Assignment No .9

Title of Assignment: MongoDB Queries:
Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations,SAVE method, logical operators etc.).

Course Objective:
To acquire the skills to use a powerful, flexible, and scalable general-purpose databases to handle Big Data

Course Outcome:

Implement NoSQL queries using MongoDB

Software Required: - Mongodb

The use Command

use DATABASE_NAME

MongoDB **use DATABASE_NAME** is used to create database. The command will create a new database, if it doesn't existotherwise it will return the existing database.**Syntax:**

Example: If you want to create a database with name **<mydb>**, then **use DATABASE** statement would be as follows:

>use mydb

To check your currently selected database use the command **db**
>db;

The dropData

MongoDB **db.dropDatabase()** command is used to drop a existing database.

Syntax: Basic syntax of **dropDatabase()** command is as follows:

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database

MongoDB Create Collection The createCollection() Method:

MongoDB **db.createCollection(name, options)** is used to create collection.

db.createCollection(name, options)

Syntax:

In the command, **name** is name of collection to be created. **Options** is a document and used to specify configuration of collection

The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

db.COLLECTION_NAME.drop()

Syntax:

MongoDB - Insert Document The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()**method.

Syntax

Basic syntax of **insert()** command is as follows:
>db.COLLECTION_NAME.insert(document)

Example

```
>db.stud1.insert({name:'abc',age:20})
```

Query Documents (Find)

In MongoDB, the db.collection.find() method retrieves documents from a collection. The db.collection.find() method returns a *cursor* to the retrieved documents. The db.collection.findOne() method also performs a read operation to return a single document. Internally, the db.collection.findOne() method is the db.collection.find() method with a limit of

1. **Select All Documents in a Collection**

An empty query document ({}) selects all documents in the collection:

```
db.inventory.find( {} )
```

Not specifying a query document to the find() is equivalent to specifying an empty query document. Therefore the following operation is equivalent to the previous operation:

```
db.inventory.find()
```

Specify Equality Condition

To specify equality condition, use the query document { <field>: <value> } to select all documents that contain the <field> with the specified <value>. The following example retrieves from the inventory collection all documents where the type field has the value snacks:

```
db.inventory.find( { type: "snacks" } )
```

Specify Conditions Using Query Operators

A query document can use the *query operators* to specify conditions in a MongoDB query.
The following example selects all documents in the inventory collection where the value of the type field is either 'food' or 'snacks':

```
db.inventory.find( { type: { $in: [ 'food', 'snacks' ] } } )
```

Although you can express this query using the $or operator, use the $in operator rather than the $or operator when performing equality checks on the same field.

Comparison Operators and Logical Operators : For comparison of different BSON type values, see the specified BSON comparison order.

Comparison	
OperatorName	**Description**
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical	
Operators Name	**Description**
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.

$not	Inverts the effect of a query expression and returns documents that do not match the query expression.

$nor Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

SAVE METHOD
The save() method has the following form:

db.collection.save(<document>, { writeConcern:

<document> })The products collection contains the

following document:

{ "_id" : 100, "item" : "water", "qty" : 30 }
The save() method performs an update with upsert:true since the document contains an

_id field:db.products.save({ _id : 100, item : "juice" })

MongoDB Logical Query Operator - $and & $not
Logical Operator - $and

The MongoDB $and operator performs a logical AND operation on an array of two or more expressions and retrieves the documents which satisfy all the expressions in the array. The $and operator uses short-circuit evaluation. If the first expression (e.g. <expression1>) evaluates to false, MongoDB will not evaluate the remaining expressions.

Syntax:
{ $and: [{ <exp1> }, { <exp2> } , ... , { <expN> }] }

Our database name is 'myinfo' and our collection name is 'student'. Here, is the

collection bellow.Example of MongoDB Logical Operator - $and

If we want to select all documents from the collection "student" which satisfying the condition -

1. sex of student is Female and
2. class of the student is VI and
3. grd_point of the student is greater than equal to 31 the following mongodb command can be used :
>db.student.find({$and:[{"sex":"Male"},{"grd_point":{ $gte: 31 }},{"class":"VI"}]}).pretty();

Logical Operator - $not
The MongoDB $not operator performs a logical NOT operation on the given expression and fetches selected documentsthat do not match the expression and the document that do not contain the field as well, specified in the expression.

Syntax:
{ field: { $not: { <expression> } } }

Example of MongoDB Logical Operator - $not
If we want to select all documents from the collection "student" which satisfying the condition -age of the student is at least 12 the following mongodb command can be used :

>db.student.find({"age": { $not: {$lt :

12}}}).pretty();The following mongodb

command can be used :

>db.student.find({"sex": { $not: /^M.*/}}).pretty();

Conclusion: We have implemented CRUD operations using Mongodb

Activity to be Submitted by Students

1. Collection “orderinfo“ which contains the documents given as below(Perform on Mongo Terminal)

{
cust_id:123
cust_name:”abc”,
status:”A”,
price:250
}

i. find the average price for each customers having status 'A'
ii. Display the status of the customers whose amount/price lie between 100 and 1000
iii. Display the customers information without “_id” .
iv. create a simple index on onderinfo collection and fire the queries.

2. Collection “movies“ which contains the documents given as below(Perform on Mongo Terminal)

{
name: “Movie1”,
type: “action”,
budget:1000000
producer:{

name: “producer1”,
address:”PUNE”
}
}

i. Find the name of the movie having budget greater than 1,00,000.
ii. Find the name of producer who lives in Pune
iii. Update the type of movie “action” to “horror”
iv. Find all the documents produced by name “producer1” with their address