2. Execute at least 10 queries on any suitable MongoDB database that demonstrates following querying techniques:

- **find and findOne (specific values)**
- **Query criteria (Query conditionals, OR queries, $not, Conditional semantics)**
- **Type-specific queries (Null, Regular expression, Querying arrays)**

`db.collection.findOne(`*query, projection*`)`

Returns one document that satisfies the specified query criteria on the collection or view. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. In capped collections, natural order is the same as insertion order. If no document satisfies the query, the method returns null.

Parameter	Type	Description
`query`	document	Optional. Specifies query selection criteria using query operators.
`projection`	document	Optional. Specifies the fields to return using projection operators. Omit this parameter to return all fields in the matching document.

```
db.orders.insertMany([{cust_id:"A123",amount:500,status:"A"},
{cust_id:"A123",amount:250,status:"A"},{cust_id:"B212",amount:200,status:"A"},
{cust_id:"A123",amount:300,status:"D"}])

> db.orders.findOne()
> db.orders.find()
```

`$and`	Joins query clauses with a logical `AND` returns all documents that match the conditions of both clauses.
`$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.

```
db.orders.find( { $and: [ { cust_id: { $ne: “A123”} }, { amount: { $exists: true } } ] } )

db.orders.find( { amount: { $ne:500 , $exists: true } } )

db.orders.find( {
   $and : [
      { $or : [ { amount: 500 }, { amount: 300 } ] },
      { $or : [ { cust_id: true }, { status: “A”} ] }
   ]
} )
```

```
db.orders.find( { amount: { $not: { $gt: 300 } } } )

db.orders.find({$nor:[ {amount:500 }, { status: true } ]  } )
```

```
db.orders.find( { $or: [ { amount: { $lt: 300 } }, { status: “A” } ] } )
```

Query for Null or Missing Fields

```
db.users.insert(
   [
      { "_id" : 900, "name" : null },
      { "_id" : 901 }
   ]
)
```

```
db.users.find( { name: null } )
```

$regex

```
db.products.insertMany([{ "_id" : 100, "sku" : "abc123", "description" : "Single line description." }

{ "_id" : 101, "sku" : "abc789", "description" : "First line\nSecond line" }

{ "_id" : 102, "sku" : "xyz456", "description" : "Many spaces before     line" }

{ "_id" : 103, "sku" : "xyz789", "description" : "Multiple\nline description" }])
```

Perform a LIKE Match

The following example matches all documents where the sku field is like "%789":

```
db.products.find( { sku: { $regex: /789$/ } } )
```

i.e . SELECT * FROM products WHERE sku like "%789";

Perform Case-Insensitive Regular Expression Match

The following example uses the i option perform a *case-insensitive* match for documents with sku value that starts with ABC.

```
db.products.find( { sku: { $regex: /^ABC/i } } )
```

Multiline Match for Lines Starting with Specified Pattern

The following example uses the m option to match lines starting with the letter S for multiline strings:

```
db.products.find( { description: { $regex: /^S/, $options: 'm' } } )
```

Use the . Dot Character to Match New Line

The following example uses the s option to allow the dot character (i.e. .) to match all characters including new line as well as the i option to perform a case-insensitive match:

```
db.products.find( { description: { $regex: /m.*line/, $options: 'si' } } )
```

Querying arrays

`$all`	Matches arrays that contain all elements specified in the query.
`$elemMatch`	Selects documents if element in the array field matches all the specified `$elemMatch` conditions.
`$size`	Selects documents if the array field is a specified size.

```
db.inventory.insertMany([

   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },

   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },

   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },

   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },

   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }

]);

>db.inventory.find( { tags: ["red", "blank"] } )

>db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```

Query for an Array Element that Meets Multiple Criteria

Use $elemMatch operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

The following example queries for documents where the dim_cm array contains at least one element that is both greater than ($gt) 22 and less than ($lt) 30:

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

Query an Array by Array Length

Use the $size operator to query for arrays by number of elements. For example, the following selects documents where the array tags has 3 elements.

```
db.inventory.find( { "tags": { $size: 3 } } )
```