0. Database structure

*: required attribute

The attribute names are different from the below description.

Collection: <Customers>

*First name
*Last name
*Email id
*Password
*Object_id (Unique, auto-generated by MongoDB)
*status
*createdOn
updatedOn

Collection: <Job>

*Object_id (Unique, auto-generated by MongoDB)
*Workers
*Cores
*Memory
*Source_files
*input_files
*Status
*IP_address
Start_time
End_Time
Logs
*CustomerID : Ref to customer
resource [id, Price] : Ref to resource
*createdOn
updatedOn
*createdBy
updatedBy

Collection: <Resources>

*Object_id (Unique, auto-generated by MongoDB)
*IP address

*Ram
*Cores
CPUs
GPUs
*Status
*Price
*Active_from
Active_to
owner : Ref to customer
*createdOn
updatedOn
*createdBy
updatedBy

1.Create Customer Collection

```
use ShareResources
```

```
db.createCollection("customer", { validator: { $jsonSchema: { bsonType:
"object", required: [ "firstname", "lastname", "emailid", "password", "createdOn",
"status"], properties: { firstname: { bsonType: "string", description: "must be a
string and is required" }, lastname: { bsonType: "string", description: "must be a
string and is required" }, emailid: { bsonType: "string", description: "must be a
string and is required" }, password: { bsonType: "string", description: "must be
a string and is required" }, createdOn: { bsonType: "date", description: "must be
a date and is required" }, updatedOn: { bsonType: "date", description: "must be
a date and is not required" }, status: { bsonType: "string", description: "must be
a string and it should be either Active, InActive or HDFS_InActive and is
required" } } } } })
```

2. Insert Customer

```
db.customer.insertOne({firstname:"Haritha",lastname:"Munagala",emailid:"mhari
tha@pdx.edu",password:"Passw0rd", createdOn:new Date(Date.now()),
status:"Active"})
```

3. Show the result

```
db.customer.find()
```

Result:

{ "_id" : ObjectId("5ababf6de538724ad170b528"), "firstname" : "Haritha", "lastname" : "Munagala", "emailid" : "mharitha@pdx.edu", "password" : "Passw0rd", "createdOn" : ISODate("2018-03-27T22:02:21.946Z"), "status" : "Active" }

4. Create Resource Collection

db.createCollection("resources", { validator: { $jsonSchema: { bsonType: "object", required: ["ip_address", "ram", "cores", "status" ,"price", "active_from", "createdOn", "createdBy"], properties: { ip_address: { bsonType: "string", description: "must be a string and is required" }, ram: { bsonType: "int", description: "must be an int and is not required" }, cores: { bsonType: "int", description: "must be an int and is required" }, cpus: { bsonType: "int", description: "must be an int" }, gpus: { bsonType: "int", description: "must be an int" }, status: { bsonType: "string", description: "must be a string and it should be either Offline or Online and is required" }, price: { bsonType: "int", description: "must be an int and is required" }, active_from: { bsonType: "date", description: "must be a date and is required" }, active_to: { bsonType: "date", description: "must be a date" }, owner: { bsonType: "objectId", description: "must be an objectId" }, createdOn: { bsonType: "date", description: "must be a date and is required" }, updatedOn: { bsonType: "date", description: "must be a date and is not required" }, createdBy: { bsonType: "objectId", description: "must be an objectId and is required" }, updatedBy: { bsonType: "objectId", description: "must be an objectId and is not required" }}}}})

5. Insert Resource - WriteError: Document failed validation

db.resources.insertOne({ ip_address: "10.456.345.566", ram: NumberInt(2), cores: NumberInt(2), status: "Offline", price: NumberInt(15), active_from: new Date(Date.now()), createdOn: new Date(Date.now()), createdBy: ObjectId("5abab56ee538724ad170b523") })

6. Show Resource

```
db.resources.find()
```

```
Result->
```

```
{ "_id" : ObjectId("5abb10a2e538724ad170b534"), "ip_address" :
"10.456.345.566", "ram" : 2, "cores" : 2, "status" : "Offline",
"price" : 15, "active_from" : ISODate("2018-03-28T03:48:50.766Z"),
"createdOn" : ISODate("2018-03-28T03:48:50.766Z"), "createdBy" :
ObjectId("5abab56ee538724ad170b523") }
```

7. Create Job Collection

```
db.createCollection("job", { validator: { $jsonSchema: { bsonType: "object",
required: ["ip_address", "workers", "cores", "memory",  "source_files",
"input_files", "status", "customerId", "resources", "createdOn", "createdBy"],
properties: { ip_address: { bsonType: "string", description: "must be a string
and is required"}, workers: { bsonType: "int", description: "must be an int and is
required"}, cores: { bsonType: "int", description: "must be an int and is
required"}, memory: { bsonType: "int", description: "must be an int and is
required" }, source_files: { bsonType: "string", description: "must be a string
and is required" }, input_files: { bsonType: "string", description: "must be a
string and is required" }, status: { bsonType: "string", description: "must be a
string and it should be either Offline or Online and is required" }, logs:
{ bsonType: "string", description: "must be a string" }, start_time: { bsonType:
"date", description: "must be a date" }, end_time: { bsonType: "date",
description: "must be a date" }, customerId: { bsonType: "objectId", description:
"must be an objectId" }, resources: { bsonType: ["array"], items: { properties:
{resourceId: { bsonType: "objectId" }, resourcePrice: { bsonType: "int" } }} },
createdOn: { bsonType: "date", description: "must be a date and is required" },
updatedOn: { bsonType: "date", description: "must be a date and is not
required" }, createdBy: { bsonType: "objectId", description: "must be an objectId
and is required" }, updatedBy: { bsonType: "objectId", description: "must be an
objectId and is not required" }}}}})
```

8. Insert Job - WriteError: Document failed validation

```
db.job.insertOne({workers: NumberInt(2), cores: NumberInt(3), memory:
NumberInt(100), source_files: "1.js", input_files: "2.js", status: "Online",
ip_address: "10.444.345.566", customerId:
ObjectId("5abab56ee538724ad170b523"), resources: [{resourceId:
ObjectId("5abab56ee538724ad170b523"), resourcePrice: NumberInt(10)},
{resourceId: ObjectId("5abab76ee538724ad170b523"), resourcePrice:
NumberInt(15)}], createdOn: new Date(Date.now()), createdBy:
ObjectId("5abab56ee538724ad170b523") })
```

9. Show Job

```
db.job.find()
```

Result ->

```
{ "_id" : ObjectId("5abb10dae538724ad170b535"), "workers" : 2, "cores"
: 3, "memory" : 100, "source_files" : "1.js", "input_files" : "2.js",
"status" : "Online", "ip_address" : "10.444.345.566", "customerId" :
ObjectId("5abab56ee538724ad170b523"), "resources" : [ { "resourceId" :
ObjectId("5abab56ee538724ad170b523"), "resourcePrice" : 10 },
{ "resourceId" : ObjectId("5abab76ee538724ad170b523"), "resourcePrice"
: 15 } ], "createdOn" : ISODate("2018-03-28T03:49:46.942Z"),
"createdBy" : ObjectId("5abab56ee538724ad170b523") }
```

10. Update customer - 1) add, delete, read by keys

1) Update

```
db.customer.update(
    { "_id" : ObjectId("5ababf6de538724ad170b528")},
    {
      $set: { "password": "Passw4rd" },
      $currentDate: { "updatedOn": true }
    }
)
```

```
db.customer.find()
```

```
{ "_id" : ObjectId("5ababf6de538724ad170b528"), "firstname" :
"Haritha", "lastname" : "Munagala", "emailid" : "mharitha@pdx.edu",
"password" : "Passw4rd", "createdOn" :
ISODate("2018-03-27T22:02:21.946Z"), "status" : "Active",
"updatedOn" : ISODate("2018-03-28T08:32:08.301Z") }
```

2) Delete

```
db.customer.remove( { "_id" : ObjectId("5ababf6de538724ad170b528") } )
db.customer.remove( { "status" : "Active" } )
```

3) Read by keys

```
db.customer.find({ "_id" : ObjectId("5abb54b6e538724ad170b536")})
```

11. Update resources

```
db.resources.update(
    { "_id" : ObjectId("5abb10a2e538724ad170b534")},
    {
      $set: { "updatedBy": ObjectId("5abb10a2e538724ad170b534") },
      $currentDate: { "updatedOn": true }
    }
)
```

```
db.resources.find()

{ "_id" : ObjectId("5abb10a2e538724ad170b534"), "ip_address" :
"10.456.345.566", "ram" : 2, "cores" : 2, "status" : "Offline",
"price" : 15, "active_from" : ISODate("2018-03-28T03:48:50.766Z"),
"createdOn" : ISODate("2018-03-28T03:48:50.766Z"), "createdBy" :
ObjectId("5abab56ee538724ad170b523"), "updatedBy" :
ObjectId("5abb10a2e538724ad170b534"), "updatedOn" :
ISODate("2018-03-28T08:46:29.327Z") }
```

12. Update job

```
db.job.find()

{ "_id" : ObjectId("5abb10dae538724ad170b535"), "workers" : 2, "cores"
: 3, "memory" : 100, "source_files" : "1.js", "input_files" : "2.js",
"status" : "Online", "ip_address" : "10.444.345.566", "customerId" :
ObjectId("5abab56ee538724ad170b523"), "resources" : [ { "resourceId" :
ObjectId("5abab56ee538724ad170b523"), "resourcePrice" : 10 },
{ "resourceId" : ObjectId("5abab76ee538724ad170b523"), "resourcePrice"
: 15 } ], "createdOn" : ISODate("2018-03-28T03:49:46.942Z"),
"createdBy" : ObjectId("5abab56ee538724ad170b523") }

db.job5.update({ "_id" : ObjectId("5abb10dae538724ad170b535") },
{ $set: { "resources" : [{ "resourceId" :
ObjectId("5abb10a2e538724ad170b534"), "resourcePrice" :
NumberInt(20) }] } })
```

13. Delete collections

```
    1) db.job.drop()
    2) db.customer.drop()
    3) db.resources.drop()
```