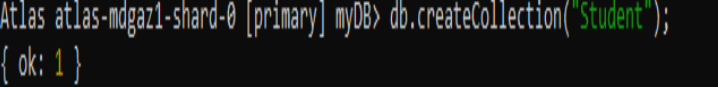
**WEEK-1**

I. Perform the following DB operations using MongoDB.

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-

Id.

**ANS-db.createCollection("Student");**



2. Insert appropriate values

**ANS-db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});**

**db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});**

**db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});**

**db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});**

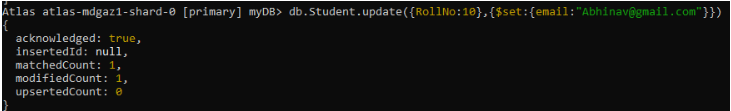
**db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"**[**rekha.de9@gmail.com**](mailto:rekha.de9@gmail.com)**"});**

**db.Student.find()**



3. Write query to update Email-Id of a student with rollno 10.

**ANS-db.Student.update({RollNo:10},{$set:{email:"**[**Abhinav@gmail.com**](mailto:Abhinav@gmail.com)**"}})**



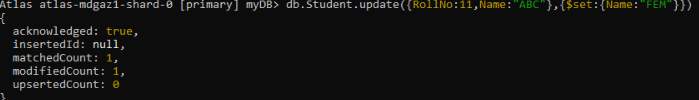
4. . Replace the student name from “ABC” to “FEM” of rollno 11

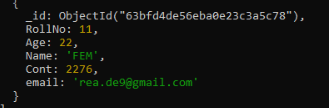
**ANS-db.Student.insert({RollNo:11,Age:22,Name:**

**"ABC",Cont:2276,email:"**[**rea.de9@gmail.com**](mailto:rea.de9@gmail.com)**"});**



**db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})**





5. Display Student Name and grade(Add if grade is not present)where the \_id column is 1.

**db.students.aggregate([**

**{ $match: { \_id: 1 } },**

**{ $project: { Name: 1, grade: { $ifNull: ["$grade", "A"] } } }**

**])**

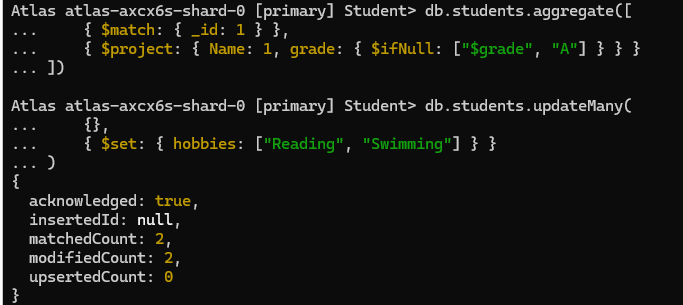
6. Update to add hobbies

**db.students.updateMany(**

**{},**

**{ $set: { hobbies: ["Reading", "Swimming"] } }**

**)**



7. Find documents where hobbies is set neither to Chess nor to Skating

**db.students.find({ hobbies: { $nin: ["Chess", "Skating"] } })**

8.Find documents whose name begins with A

**db.students.find({ Name: /^A/ })**

II. Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes.

Cust\_id, Acc\_Bal, Acc\_Type

**ANS-1.db.createCollection(“Customer”);**

2. Insert at least 5 values into the table

**ANS-2.db.Customers.insert({Cust\_id:1,Acc\_Bal:2000,Acc\_Type:"Savings"});**

**db.Cusotmers.insert({Cust\_id:2,Acc\_Bal:3000,Acc\_Type:"Savings"});**

**db.Customers.insert({Cust\_id:3,Acc\_Bal:1500,Acc\_Type:"Savings"});**

**db.Customers.insert({Cust\_id:4,Acc\_Bal:1000,Acc\_Type:"Current"});**

**db.Customers.insert({Cust\_id:5,Acc\_Bal:2000,Acc\_Type:"Current"});**



3. Write a query to display those records whose total account balance is greater than

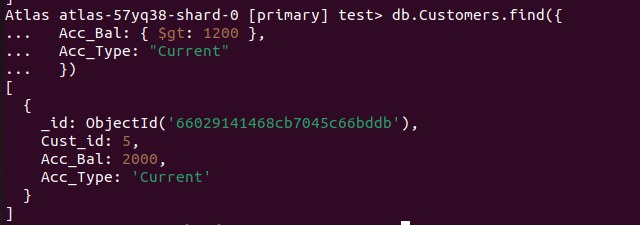
1200 of account type ‘Z’ for each customer\_id.

**ANS-3. db.Customers.find({**

**Acc\_Bal: { $gt: 1200 },**

**Acc\_Type: "Current"**

**})**



4. Determine Minimum and Maximum account balance for each customer\_i

**ANS-4 db.Customers.aggregate([ { $group: { \_id: "$Cust\_id", min\_balance: { $min: "$Acc\_Bal" }, max\_balance: { $max: "$Acc\_Bal" } } }] )**

5. Sort the documents based on Customer ID in ascending order and Account Balance in descending order

**db.Customers.aggregate([**

**{**

**$match: {**

**Acc\_Type: "Current"**

**}**

**},**

**{**

**$group: {**

**\_id: "$customer\_id",**

**min\_balance: { $min: "$Acc\_Bal" },**

**max\_balance: { $max: "$Acc\_Bal" }**

**}**

**},**

**{**

**$sort: {**

**"\_id": 1,**

**"max\_balance": -1**

**}**

**}**

**])**



6.Display only 2nd and 3rd records from the collection

**db.Customers.find().skip(1).limit(2)**

