BDA Lab 3 Work

Q1. Perform the following DB operations using MongoDB.

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
```

2. Insert appropriate values

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bca6c18a3d93e29d14a0e') }
}
Atlas atlas-8o67tr-shard-0 [primary] test>

Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bca6c18a3d93e29d14a0f') }
}
Atlas atlas-8o67tr-shard-0 [primary] test>
https://docs.mongodb.com/mongodb-shell/
Ctrl+Click to follow link

Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bca6c18a3d93e29d14a10') }
}
Atlas atlas-8o67tr-shard-0 [primary] test>

Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bca6c18a3d93e29d14a11') }
}
Atlas atlas-8o67tr-shard-0 [primary] test>

Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bca9118a3d93e29d14a12') }
}
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.update({RollNo:10},{$set:{
... email:"Abhinav@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.insert({RollNo:11,Age:22,Name:
... "ABC",Cont:2276,email:"rea.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bcab918a3d93e29d14a13') }
}
```

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.students.insertMany([
...     { "_id": 1, "Rollno": 10, "Age": 20, "ContactNo": "1234567890", "Email-Id": "student1@example.com", "Name": "John", "Grade": "A" },
...     { "_id": 2, "Rollno": 11, "Age": 21, "ContactNo": "1234567891", "Email-Id": "student2@example.com", "Name": "ABC", "Grade": "B" },
...     { "_id": 3, "Rollno": 12, "Age": 22, "ContactNo": "1234567892", "Email-Id": "student3@example.com", "Name": "Jane", "Grade": "B" }
... ])
```

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.students.find({ "_id": 1 }, { "Name": 1, "Grade": 1 })
[ { _id: 1, Name: 'John', Grade: 'A' } ]
```

6. Update to add hobbies

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.students.updateMany(
...     {},
...     { $set: { "Hobbies": ["Reading", "Sports"] } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.students.find({ "Hobbies": { $nin: ["Chess", "Skating"] } })
[
  {
    _id: 1,
    Rollno: 10,
    Age: 20,
    ContactNo: '1234567890',
    'Email-Id': 'student1@example.com',
    Name: 'John',
    Grade: 'A',
    Hobbies: [ 'Reading', 'Sports' ]
  },
  {
    _id: 2,
    Rollno: 11,
    Age: 21,
    ContactNo: '1234567891',
    'Email-Id': 'student2@example.com',
    Name: 'ABC',
    Grade: 'B',
    Hobbies: [ 'Reading', 'Sports' ]
  },
  {
    _id: 3,
    Rollno: 12,
    Age: 22,
    ContactNo: '1234567892',
    'Email-Id': 'student3@example.com',
    Name: 'Jane',
    Grade: 'B',
    Hobbies: [ 'Reading', 'Sports' ]
  }
]
```

8. Find documents whose name begins with A

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.students.find({ "Name": /^A/ })
[
  {
    _id: 2,
    Rollno: 11,
    Age: 21,
    ContactNo: '1234567891',
    'Email-Id': 'student2@example.com',
    Name: 'ABC',
    Grade: 'B',
    Hobbies: [ 'Reading', 'Sports' ]
  }
]
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.createCollection("Customers")
{ ok: 1 }
```

2. Insert at least 5 values into the table

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Customers.insertMany([
...     { "Cust_id": 1, "Acc_Bal": 1000, "Acc_Type": "Z" },
...     { "Cust_id": 1, "Acc_Bal": 500, "Acc_Type": "X" },
...     { "Cust_id": 2, "Acc_Bal": 1500, "Acc_Type": "Z" },
...     { "Cust_id": 3, "Acc_Bal": 2000, "Acc_Type": "Z" },
...     { "Cust_id": 3, "Acc_Bal": 700, "Acc_Type": "Y" },
...     { "Cust_id": 4, "Acc_Bal": 800, "Acc_Type": "Z" },
...     { "Cust_id": 5, "Acc_Bal": 1300, "Acc_Type": "Z" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660bcbfb18a3d93e29d14a14'),
    '1': ObjectId('660bcbfb18a3d93e29d14a15'),
    '2': ObjectId('660bcbfb18a3d93e29d14a16'),
    '3': ObjectId('660bcbfb18a3d93e29d14a17'),
    '4': ObjectId('660bcbfb18a3d93e29d14a18'),
    '5': ObjectId('660bcbfb18a3d93e29d14a19'),
    '6': ObjectId('660bcbfb18a3d93e29d14a1a')
  }
}
```

3. Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer_id.

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Customers.aggregate([
...     { $match: { "Acc_Type": "Z" } },
...     { $group: { "_id": "$Cust_id", "Total_Balance": { $sum: "$Acc_Bal" } } },
...     { $match: { "Total_Balance": { $gt: 1200 } } }
... ])
[
  { _id: 5, Total_Balance: 1300 },
  { _id: 3, Total_Balance: 2000 },
  { _id: 2, Total_Balance: 1500 }
]
```

4. Determine Minimum and Maximum account balance for each customer

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Customers.aggregate([
...     { $group: { "_id": "$Cust_id", "Min_Balance": { $min: "$Acc_Bal" }, "Max_Balance": { $max: "$Acc_Bal" } } }
... ])
[
  { _id: 4, Min_Balance: 800, Max_Balance: 800 },
  { _id: 1, Min_Balance: 500, Max_Balance: 1000 },
  { _id: 2, Min_Balance: 1500, Max_Balance: 1500 },
  { _id: 3, Min_Balance: 700, Max_Balance: 2000 },
  { _id: 5, Min_Balance: 1300, Max_Balance: 1300 }
]
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Customers.find().sort({ "Cust_id": 1, "Acc_Bal": -1 })
[
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a14'),
    Cust_id: 1,
    Acc_Bal: 1000,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a15'),
    Cust_id: 1,
    Acc_Bal: 500,
    Acc_Type: 'X'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a16'),
    Cust_id: 2,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a17'),
    Cust_id: 3,
    Acc_Bal: 2000,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a18'),
    Cust_id: 3,
    Acc_Bal: 700,
    Acc_Type: 'Y'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a19'),
    Cust_id: 4,
    Acc_Bal: 800,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a1a'),
    Cust_id: 5,
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  }
]
```

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

```
Atlas atlas-8o67tr-shard-0 [primary] test> db.Customers.find().skip(1).limit(2)
[
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a15'),
    Cust_id: 1,
    Acc_Bal: 500,
    Acc_Type: 'X'
  },
  {
    _id: ObjectId('660bcbfb18a3d93e29d14a16'),
    Cust_id: 2,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  }
]
```

III. Perform the following DB operations using MongoDB
Create a collection by the name blogPosts and it has 3 fields id, title and comments. In the collection the comments field is an array which consists of user details. Each collection
consists of two user details inside the comments array- user name and text
Demonstrate the following

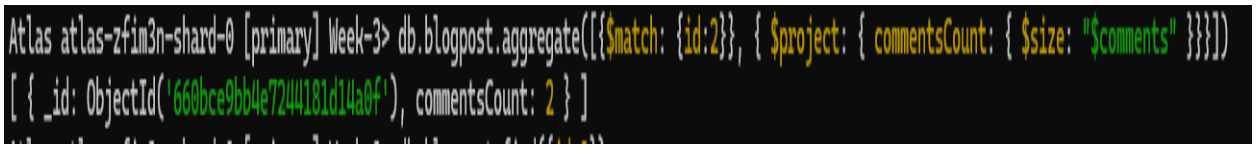
1. Creating 2 collections

```
C:\Users\student>mongosh "mongodb+srv://cluster0.99a58av.mongodb.net/" --apiVersion 1 --username likhithcs21
Enter password: **********
Current Mongosh Log ID: 660bcb61b4e7244181d14a0d
Connecting to:          mongodb+srv://<credentials>@cluster0.99a58av.mongodb.net/?appName=mongosh+2.2.1
Using MongoDB:          7.0.7 (API Version 1)
Using Mongosh:          2.2.1
mongosh 2.2.2 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-zfim3n-shard-0 [primary] test> use Week-3
switched to db Week-3
Atlas atlas-zfim3n-shard-0 [primary] Week-3> db.createCollection('blogpost')
{ ok: 1 }
Atlas atlas-zfim3n-shard-0 [primary] Week-3> db.blogpost.insert({id:1,title:'Abc',comments:[{userName:'Likhith',text:'Hello'},{userName:'Ramesh',text:
'Hii'}]})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660bcd36b4e7244181d14a0e') }
Atlas atlas-zfim3n-shard-0 [primary] Week-3> db.blogpost.insertOne({id:2,title:'bba',comments:[{userName:'Likhith',text:'It is good'},{userName:'Sures
h',text:'Intersting'}]})
{
  acknowledged: true,
  insertedId: ObjectId('660bce9bb4e7244181d14a0f')
}
```

2. Display second element (id = 2)

```
Atlas atlas-zfim3n-shard-0 [primary] Week-3> db.blogpost.find({id:2})
[
  {
    _id: ObjectId('660bce9bb4e7244181d14a0f'),
    id: 2,
    title: 'bba',
    comments: [
      { userName: 'Likhith', text: 'It is good' },
      { userName: 'Suresh', text: 'Intersting' }
    ]
  }
]
```

3 .Display size of the array



4. Display first two elements of the array

5. Update the document with id 4 and replace the element present in 1st index position of the array with another array