Commands

use *databasename*

db.createCollection(“tablename”) //say our table name is s

db.s.insertOne({Attribute1: value, Attribute2: value})

db.s.insertMany([{Attribute1: value, Attribute2: value}, {Attribute1: value, Attribute2: value}])

db.s.find() //Show all

show collections //list the tables

db.s.updateOne({Attribute1: “findvalue”}, {$set: {Attribute1: “newvalue”}}, {upsert: true})

db.s.updateMany({}, {$set: {NewAttribute: “Value”}}, {upsert: true}) //update all entries

db.c.find({Attribute1: {$gt: xxx(value)}, Attribute2: “value”}) //greater than

db.c.aggregate([{"$group": {"_id": null ,"max": {"$max": "$Acc_Bal"}, "min": {"$min":"$Acc_Bal" }}}])

db.c.aggregate([{"$group": {"_id": "Cust_id" ,"max": {"$max": "$Acc_Bal"}, "min": {"$min":"$Acc_Bal" }}}])

I. Perform the following DB operations using MongoDB.

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.
2. Insert appropriate values
3. Write query to update Email-Id of a student with rollno 10.
4. . Replace the student name from “ABC” to “FEM” of rollno 11

```
Student> db.s.insertMany([{StudName:"JohnDoe", Grade:"VIII", Hobbies:"Painting"},
... {StudName:"SarahSmith", Grade:"VI", Hobbies:"Reading"},
... {StudName:"DavidWilliams", Grade:"IX", Hobbies:"Coding"},
... {StudName:"EmilyJones", Grade:"V", Hobbies:"Dancing"},
... {StudName:"OliverBrown", Grade:"X", Hobbies:"Photography"},
... {StudName:"AvaGarcia", Grade:"IV", Hobbies:"Writing"},
... {StudName:"WilliamParker", Grade:"VII", Hobbies:"Chess"},
... {StudName:"OliviaWilson", Grade:"III", Hobbies:"Cooking"},
... {StudName:"MichaelHall", Grade:"XI", Hobbies:"Gardening"},
... {StudName:"CharlotteRoberts", Grade:"II", Hobbies:"Swimming"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660a7cce2d86dcfefdc00e80'),
    '1': ObjectId('660a7cce2d86dcfefdc00e81'),
    '2': ObjectId('660a7cce2d86dcfefdc00e82'),
    '3': ObjectId('660a7cce2d86dcfefdc00e83'),
    '4': ObjectId('660a7cce2d86dcfefdc00e84'),
    '5': ObjectId('660a7cce2d86dcfefdc00e85'),
    '6': ObjectId('660a7cce2d86dcfefdc00e86'),
    '7': ObjectId('660a7cce2d86dcfefdc00e87'),
    '8': ObjectId('660a7cce2d86dcfefdc00e88'),
    '9': ObjectId('660a7cce2d86dcfefdc00e89')
  }
}
Student> db.s.find()
[
  {
    _id: ObjectId('660a7a7e2d86dcfefdc00e7e'),
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternsetSurfing'
  },
  {
    _id: ObjectId('660a7ab22d86dcfefdc00e7f'),
    StudName: 'Aryana',
    Grade: 'VIII',
    Hobbies: 'InternsetSurfing'
  },
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e80'),
    StudName: 'JohnDoe',
    Grade: 'VIII',
    Hobbies: 'Painting'
  },
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e81'),
    StudName: 'SarahSmith',
    Grade: 'VI',
    Hobbies: 'Reading'
  },
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e82'),
    StudName: 'DavidWilliams',
    Grade: 'IX',
```

```
    Grade: 'III',
    Hobbies: 'Cooking'
  },
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e88'),
    StudName: 'MichaelHall',
    Grade: 'XI',
    Hobbies: 'Gardening'
  },
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e89'),
    StudName: 'CharlotteRoberts',
    Grade: 'II',
    Hobbies: 'Swimming'
  }
]
Student> db.s.upodateOne({StudName:"AvaGarcia"}, {$set:{StudName: 'Ambika'}})
TypeError: db.s.upodateOne is not a function
Student> db.s.updateOne({StudName:"AvaGarcia"}, {$set:{StudName: 'Ambika'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Student> db.s.find({StudName:'Ambika'})
[
  {
    _id: ObjectId('660a7cce2d86dcfefdc00e85'),
    StudName: 'Ambika',
    Grade: 'IV',
    Hobbies: 'Writing'
  }
]
Student> db.s.updateOne({StudName:"Ada"}, {$set:{StudName: 'Ambuja'}}, {upsert: true})
{
  acknowledged: true,
  insertedId: ObjectId('660a7e7c7b7d50080592d516'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
Student> db.s.find({StudName:'Ambuja'})
[ { _id: ObjectId('660a7e7c7b7d50080592d516'), StudName: 'Ambuja' } ]
Student> db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG'}, {Hobbies: 'Staring at Wall'}}, {upsert: true})
Uncaught:
SyntaxError: Unexpected token (1:58)

> 1 | db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG'}, {Hobbies: 'Staring at Wall'}}, {upsert: true})
    |                                                           ^
  2 |

Student> db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG', Hobbies: 'Staring at Wall'}}, {upsert: true})
{
```

```
  }
]
Student> db.s.updateOne({StudName:"Ada"}, {$set:{StudName: 'Ambuja'}}, {upsert: true})
{
  acknowledged: true,
  insertedId: ObjectId('660a7e7c7b7d50080592d516'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
Student> db.s.find({StudName:'Ambuja'})
[ { _id: ObjectId('660a7e7c7b7d50080592d516'), StudName: 'Ambuja' } ]
Student> db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG'}, {Hobbies: 'Staring at Wall'}}, {upsert: true})
Uncaught:
SyntaxError: Unexpected token (1:58)

> 1 | db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG'}, {Hobbies: 'Staring at Wall'}}, {upsert: true})
    |                                                          ^
  2 |

Student> db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG', Hobbies: 'Staring at Wall'}}, {upsert: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Student> db.s.find({StudName:'Ambuja'})
[
  {
    _id: ObjectId('660a7e7c7b7d50080592d516'),
    StudName: 'Ambuja',
    Grade: 'LKG',
    Hobbies: 'Staring at Wall'
  }
]
Student> db.s.updateOne({StudName:"Ambuja"}, {$set:{Grade: 'LKG', Hobbies: 'Chewing Gum'}}, {upsert: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Student> db.s.find({StudName:'Ambuja'})
[
  {
    _id: ObjectId('660a7e7c7b7d50080592d516'),
    StudName: 'Ambuja',
    Grade: 'LKG',
    Hobbies: 'Chewing Gum'
  }
]
Student>
```

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

2. Insert at least 5 values into the table

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

4. Determine Minimum and Maximum account balance for each customer_i

```
Student> db.createCollection("c")
{ ok: 1 }
Student> db.c.insertMany([
...     {Cust_id: "0001", Acc_Bal: 5000, Acc_type: "Savings"},
...     {Cust_id: "0002", Acc_Bal: 12000, Acc_type: "Current"},
...     {Cust_id: "0003", Acc_Bal: 7500, Acc_type: "Savings"},
...     {Cust_id: "0004", Acc_Bal: 25000, Acc_type: "Current"},
...     {Cust_id: "0005", Acc_Bal: 10000, Acc_type: "Savings"},
...     {Cust_id: "0006", Acc_Bal: 17000, Acc_type: "Current"},
...     {Cust_id: "0007", Acc_Bal: 8000, Acc_type: "Savings"},
...     {Cust_id: "0008", Acc_Bal: 32000, Acc_type: "Current"},
...     {Cust_id: "0009", Acc_Bal: 15000, Acc_type: "Savings"},
...     {Cust_id: "0010", Acc_Bal: 20000, Acc_type: "Current"}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660a84152d86dcfefdc00e8a'),
    '1': ObjectId('660a84152d86dcfefdc00e8b'),
    '2': ObjectId('660a84152d86dcfefdc00e8c'),
    '3': ObjectId('660a84152d86dcfefdc00e8d'),
    '4': ObjectId('660a84152d86dcfefdc00e8e'),
    '5': ObjectId('660a84152d86dcfefdc00e8f'),
    '6': ObjectId('660a84152d86dcfefdc00e90'),
    '7': ObjectId('660a84152d86dcfefdc00e91'),
    '8': ObjectId('660a84152d86dcfefdc00e92'),
    '9': ObjectId('660a84152d86dcfefdc00e93')
  }
}
Student>
```

```
Student> db.c.find({Acc_Bal: {$gt: 1200}, Acc_type: "Current"})
[
  {
    _id: ObjectId('660a84152d86dcfefdc00e8b'),
    Cust_id: '0002',
    Acc_Bal: 12000,
    Acc_type: 'Current'
  },
  {
    _id: ObjectId('660a84152d86dcfefdc00e8d'),
    Cust_id: '0004',
    Acc_Bal: 25000,
    Acc_type: 'Current'
  },
  {
    _id: ObjectId('660a84152d86dcfefdc00e8f'),
    Cust_id: '0006',
    Acc_Bal: 17000,
    Acc_type: 'Current'
  },
  {
    _id: ObjectId('660a84152d86dcfefdc00e91'),
    Cust_id: '0008',
    Acc_Bal: 32000,
    Acc_type: 'Current'
  },
  {
    _id: ObjectId('660a84152d86dcfefdc00e93'),
    Cust_id: '0010',
    Acc_Bal: 20000,
    Acc_type: 'Current'
  }
]
Student>
```

```
Student> db.c.aggregate([{"$group": {"_id": null ,"max": {"$max": "$Acc_Bal"}, "min": {"$min":"$Acc_Bal" }}}])
[ { _id: null, max: 32000, min: 5000 } ]
Student> db.c.aggregate([{"$group": {"_id": "Cust_id" ,"max": {"$max": "$Acc_Bal"}, "min": {"$min":"$Acc_Bal" }}}])
[ { _id: 'Cust_id', max: 32000, min: 5000 } ]
Student>
```