CS 585 Big Data Management
Project 3

Name: Muralidharan Kumaravel, Jing Yang

Problem 4 Mongo DB

Creating the collection test.

```
mongosh mongodb://127.0.0.

project3> db.createCollection("test")
{ ok: 1 }
project3>
```

Inserting initial 10 documents.

```
mongosh mongodb://127.0.0.
...                 "award" : "The Economist Innovation Award",
...                 "year" : 2002,
...                 "by" : "The Economist"
...             },
...             {
...                 "award" : "Officer of the Order of Canada",
...                 "year" : 2007,
...                 "by" : "Canada"
...             }
...         ]
...     },
...     {
...         "_id" : 10,
...         "name" : {
...             "first" : "Martin",
...             "last" : "Odersky"
...         },
...         "contribs" : [
...             "Scala"
...         ]
...     }
...
... ] )
{
  acknowledged: true,
  insertedIds: {
    '0': 1,
    '1': ObjectId("51df07b094c6acd67e492f41"),
    '2': 3,
    '3': 4,
    '4': 5,
    '5': 6,
    '6': ObjectId("51e062189c6ae665454e301d"),
    '7': 8,
    '8': 9,
    '9': 10
  }
}
project3>
```

Question 1:

1. Write a CRUD operation(s) that inserts the following new records into the collection:

```
mongosh mongodb://127.0.0.
...         },
...         "birth" : ISODate("1933-08-27T04:00:00Z"),
...         "death" : ISODate("1984-11-07T04:00:00Z"),
...         "contribs" : [
...             "C++",
...             "Simula"
...         ],
...         "awards" : [
...             {
...                 "award" : "WPI Award",
...                 "year" : 1977,
...                 "by" : "WPI"
...             }
...         ]
...     },
...     {
...         "_id" : 30,
...         "name" : {
...             "first" : "David",
...             "last" : "Mark"
...         },
...         "birth" : ISODate("1911-04-12T04:00:00Z"),
...         "death" : ISODate("2000-11-07T04:00:00Z"),
...         "contribs" : [
...             "C++",
...             "FP",
...             "Lisp"
...         ],
...         "awards" : [
...             {
...                 "award" : "WPI Award",
...                 "year" : 1963,
...                 "by" : "WPI"
...             },
...             {
...                 "award" : "Turing Award",
...                 "year" : 1966,
...                 "by" : "ACM"
...             }
...         ]
...     }
... ])
{ acknowledged: true, insertedIds: { '0': 20, '1': 30 } }
project3>
```

2. Report all documents of people who got less than 3 awards or have contributed in “FP”

```
mongosh mongodb://127.0.0.
        year: '2011',

project3> db.test.find({ $or: [{$where: "this.awards ? this.awards.length < 3 : false"},{contribs: "FP"}]});
[
  {
    _id: 1,
    name: { first: 'John', last: 'Backus' },
    birth: ISODate("1924-12-03T05:00:00.000Z"),
    death: ISODate("2007-03-17T04:00:00.000Z"),
    contribs: [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    awards: [
      {
        award: 'W.W. McDowell Award',
        year: 1967,
        by: 'IEEE Computer Society'
      },
      {
        award: 'National Medal of Science',
        year: 1975,
        by: 'National Science Foundation'
      },
      { award: 'Turing Award', year: 1977, by: 'ACM' },
      {
        award: 'Draper Prize',
        year: 1993,
        by: 'National Academy of Engineering'
      }
    ]
  },
  {
    _id: 6,
    name: { first: 'Guido', last: 'van Rossum' },
    birth: ISODate("1956-01-31T05:00:00.000Z"),
    contribs: [ 'Python' ],
    awards: [
      {
        award: 'Award for the Advancement of Free Software',
        year: 2001,
        by: 'Free Software Foundation'
      },
      { award: 'NLUUG Award', year: 2003, by: 'NLUUG' }
    ]
  },
  {
```

3. Update the document of “Guido van Rossum” to add “OOP” to the contribution list.

```
mongosh mongodb://127.0.0.

project3> db.test.updateOne({ name: {"first": "Guido", "last": "van Rossum" }}, {$push: {contribs: "OOP"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
project3> db.test.find({ name: {"first": "Guido", "last": "van Rossum" }})
[
  {
    _id: 6,
    name: { first: 'Guido', last: 'van Rossum' },
    birth: ISODate("1956-01-31T05:00:00.000Z"),
    contribs: [ 'Python', 'OOP' ],
    awards: [
      {
        award: 'Award for the Advancement of Free Software',
        year: 2001,
        by: 'Free Software Foundation'
      },
      { award: 'NLUUG Award', year: 2003, by: 'NLUUG' }
    ]
  }
]
project3>
```

4. Insert a new filed of type array, called “comments”, into the document of “Alex Chen” storing the following comments: “He taught in 3 universities”, “died from cancer”, “lived in CA”.

```
mongosh mongodb://127.0.0.

project3> db.test.updateOne({name: {"first": "Alex","last": "Chen"}}, {$set: {comments: ["He taught in 3 universities","died from cancer","lived in CA"]}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
project3> db.test.find({name: {"first": "Alex","last": "Chen"}})
[
  {
    _id: 20,
    name: { first: 'Alex', last: 'Chen' },
    birth: ISODate("1933-08-27T04:00:00.000Z"),
    death: ISODate("1984-11-07T04:00:00.000Z"),
    contribs: [ 'C++', 'Simula' ],
    awards: [ { award: 'WPI Award', year: 1977, by: 'WPI' } ],
    comments: [
      'He taught in 3 universities',
      'died from cancer',
      'lived in CA'
    ]
  }
]
project3>
```

5. For each contribution by “Alex Chen”, say X, and list the peoples’ names (first and last) who have contribution X. E.g., Alex Chen has two contributions in “C++” and “Simula”. Then, the output should be similar to: a. {Contribution: “C++”, People: [{first: “Alex”, last: “Chen”}, {first: “David”, last: “Mark”}]}, { Contribution: “Simula”, ...}

```
mongosh mongodb://127.0.0.

project3> var contri = []

project3> db.test.find({name: {"first": "Alex","last": "Chen"}}).forEach(function(x) {contri = x.contribs});

project3> db.test.aggregate([{$unwind: "$contribs"},{$match: {'contribs': {$in: contri}}},{$group: {_id: "$contribs",people: {$push: "$name"}}}]);
[
  {
    _id: 'Simula',
    people: [
      { first: 'Kristen', last: 'Nygaard' },
      { first: 'Ole-Johan', last: 'Dahl' },
      { first: 'Alex', last: 'Chen' }
    ]
  },
  {
    _id: 'C++',
    people: [
      { first: 'Alex', last: 'Chen' },
      { first: 'David', last: 'Mark' }
    ]
  }
]
project3>
```

6. Report the distinct organization that gave awards. This information can be found in the “by” field inside the “awards” array. The output should be an array of distinct values, e.g., [“wpi’, “acm’, ...]

```
mongosh mongodb://127.0.0.

project3> db.test.distinct("awards.by")
[
  ' British Computer Society',
  'ACM',
  'Canada',
  'Data Processing Management Association',
  'Free Software Foundation',
  'IEEE',
  'IEEE Computer Society',
  'Inamori Foundation',
  'NLUUG',
  'National Academy of Engineering',
  'National Science Foundation',
  'Norwegian Data Association',
  'The Economist',
  'The Japan Prize Foundation',
  'United States',
  'WPI'
]
project3>
```

7. Delete from all documents any award given in 2011

```
mongosh mongodb://127.0.0.

project3> db.test.updateMany({}, {$pull: {"awards": {year: {$in: [2011, "2011"]}}}}, {multi: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 12,
  modifiedCount: 0,
  upsertedCount: 0
}
project3>
```

8. Report only the names (first and last) of those individuals who won at least two awards in 2001.

```
mongosh mongodb://127.0.0.

project3> db.test.aggregate({$project: {"first": "$name.first","last": "$name.last","awards2001": {$size: {$filter: {input: {$ifNull: ["$awards", []]},as: "aw2001",cond: {$eq: ["$
$aw2001.year", 2001]}}}}}}, {$group: {_id: {"first": "$first","last": "$last"},awards: {$sum: "$awards2001"}}}, {$match: {awards: {$gte: 2}}})
[
  { _id: { first: 'Ole-Johan', last: 'Dahl' }, awards: 2 },
  { _id: { first: 'Kristen', last: 'Nygaard' }, awards: 2 }
]
project3>
```

9. Report the document with the largest id. First, you need to find the largest _id (using a CRUD statement), and then use that to report the corresponding document.

```
mongosh mongodb://127.0.0.

project3> db.test.find().sort({_id: -1}).limit(1)
[
  {
    _id: ObjectId("51e062189c6ae665454e301d"),
    name: { first: 'Dennis', last: 'Ritchie' },
    birth: ISODate("1941-09-09T04:00:00.000Z"),
    death: ISODate("2011-10-12T04:00:00.000Z"),
    contribs: [ 'UNIX', 'C' ],
    awards: [
      { award: 'Turing Award', year: 1983, by: 'ACM' },
      {
        award: 'National Medal of Technology',
        year: 1998,
        by: 'United States'
      }
    ]
  }
]
project3>
```

10. Report only one document where one of the awards is given by “ACM”.

```
mongosh mongodb://127.0.0.

project3> db.test.find({"awards.by" : "ACM"}).limit(1)
[
  {
    _id: 1,
    name: { first: 'John', last: 'Backus' },
    birth: ISODate("1924-12-03T05:00:00.000Z"),
    death: ISODate("2007-03-17T04:00:00.000Z"),
    contribs: [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    awards: [
      {
        award: 'W.W. McDowell Award',
        year: 1967,
        by: 'IEEE Computer Society'
      },
      {
        award: 'National Medal of Science',
        year: 1975,
        by: 'National Science Foundation'
      },
      { award: 'Turing Award', year: 1977, by: 'ACM' },
      {
        award: 'Draper Prize',
        year: 1993,
        by: 'National Academy of Engineering'
      }
    ]
  }
]
project3>
```

Question 2

1. Write an aggregation query that group by the award name, i.e., the “award” field inside the “awards” array, and reports the count of each award.

```
mongosh mongodb://127.0.0.

project3> db.test.aggregate([{"$unwind": "$awards"},{"$group": {"_id": "$awards.award","count": {$sum: 1}}}])
[
  { _id: 'Draper Prize', count: 1 },
  { _id: 'Distinguished Fellow', count: 1 },
  { _id: 'W.W. McDowell Award', count: 1 },
  { _id: 'Rosing Prize', count: 2 },
  { _id: 'Award for the Advancement of Free Software', count: 1 },
  { _id: 'Turing Award', count: 6 },
  { _id: 'WPI Award', count: 2 },
  { _id: 'IEEE John von Neumann Medal', count: 2 },
  { _id: 'W. W. McDowell Award', count: 1 },
  { _id: 'Computer Sciences Man of the Year', count: 1 },
  { _id: 'National Medal of Technology', count: 2 },
  { _id: 'National Medal of Science', count: 2 },
  { _id: 'The Economist Innovation Award', count: 1 },
  { _id: 'Kyoto Prize', count: 1 },
  { _id: 'NLUUG Award', count: 1 },
  { _id: 'Officer of the Order of Canada', count: 1 }
]
project3>
```

2. Write an aggregation query that groups by the birth year, i.e., the year within the “birth” field, and report an array of _ids for each birth year

```
mongosh mongodb://127.0.0.

project3> db.test.aggregate({$group: {_id: {$year: "$birth"},id: {$push: "$_id"}}})
[
  { _id: 1927, id: [ ObjectId("51df07b094c6acd67e492f41") ] },
  { _id: 1965, id: [ 8 ] },
  { _id: 1933, id: [ 20 ] },
  { _id: null, id: [ 10 ] },
  { _id: 1941, id: [ ObjectId("51e062189c6ae665454e301d") ] },
  { _id: 1906, id: [ 3 ] },
  { _id: 1924, id: [ 1 ] },
  { _id: 1926, id: [ 4 ] },
  { _id: 1955, id: [ 9 ] },
  { _id: 1911, id: [ 30 ] },
  { _id: 1956, id: [ 6 ] },
  { _id: 1931, id: [ 5 ] }
]
project3>
```

3. Report the document with the smallest and largest _ids. You first need to find the values of the smallest and largest and then report their documents.

```
mongosh mongodb://127.0.0.
project3> db.test.aggregate([{$group: {_id : '', max: {$max : '$_id'}}}])
[ { _id: '', max: ObjectId("51e062189c6ae665454e301d") } ]
project3> db.test.aggregate([{$group: {_id : '', min: {$min : '$_id'}}}])
[ { _id: '', min: 1 } ]
project3> db.test.aggregate([
...   { $facet: {
...     max: [{$group: {_id : '', max: {$max : '$_id'}}}],
...     min:  [{$group: {_id : '', min: {$min : '$_id'}}}]
...   }}
... ])
[
  {
    max: [ { _id: '', max: ObjectId("51e062189c6ae665454e301d") } ],
    min: [ { _id: '', min: 1 } ]
  }
]
project3>
```

Question 3

Creating the data for Q1 and Q2:

```
mongosh mongodb://127.0.0.
project3> db.question3part1.insertOne({_id:"dbm", parent:"Databases"})
{ acknowledged: true, insertedId: 'dbm' }
project3> db.question3part1.insertOne({_id:"Databases", parent:"Programming"})
{ acknowledged: true, insertedId: 'Databases' }
project3> db.question3part1.insertOne({_id:"Languages", parent:"Programming"})
{ acknowledged: true, insertedId: 'Languages' }
project3> db.question3part1.insertOne({_id:"Programming", parent:"Books"})
{ acknowledged: true, insertedId: 'Programming' }
project3> db.question3part1.insertOne({_id:"Books", parent:null})
{ acknowledged: true, insertedId: 'Books' }
project3>
```

1. Assume we model the records and relationships in Figure 1 using the Parent-Referencing model (Slide 4 in MongoDB-3). Write a query to report the ancestors of “MongoDB”. The output should be an array containing values [{Name: “Databases”, Level: 1}, {Name: “Programming”, Level: 2}, {Name: “Books”, Level: 3}] * Note: “Level” is the distance from the “MongoDB” node to the other node. It should be computed in your code

```
mongosh mongodb://127.0.0.

project3> var t1 = db.question3part1.findOne({_id: "MongoDB"});

project3> var t2 = db.question3part1.findOne({ _id: t1.parent});

project3> var l = 0;

project3> var p = [];

project3> while (t2) {
...     l++;
...     p.push({"Name": t2._id,"Level": l});
...     t2 = db.categories.findOne({_id: t2.parent});
... }
null
project3> p
[ { Name: 'Databases', Level: 1 } ]
project3>
```

2. Assume we model the records and relationships in Figure 1 using the Parent-Referencing model (Slide 4 in MongoDB-3). You are given only the root node, i.e., _id = “Books”, and write a query that reports the height of the tree. (It should be 4 in our case).

```
mongosh mongodb://127.0.0.

project3> var r = db.question3part1.findOne({parent: "Books"});

project3> var c = 1;

project3> while (r) {
...     r = db.question3part1.findOne({parent: r["_id"]});
...     c ++;
... }c
4
project3>
```

Creating the data for question 3 and 4:

```
mongosh mongodb://127.0.0.

project3> db.question3part2.insertOne({_id: "MongoDB",children: []})
{ acknowledged: true, insertedId: 'MongoDB' }
project3> db.question3part2.insertOne({_id: "dbm",children: []})
{ acknowledged: true, insertedId: 'dbm' }
project3> db.question3part2.insertOne({_id: "Databases",children: ["MongoDB", "dbm"]})
{ acknowledged: true, insertedId: 'Databases' }
project3> db.question3part2.insertOne({_id: "Languages",children: []})
{ acknowledged: true, insertedId: 'Languages' }
project3> db.question3part2.insertOne({_id: "Programming",children: ["Databases", "Languages"]})
{ acknowledged: true, insertedId: 'Programming' }
project3> db.question3part2.insertOne({_id: "Books",children: ["Programming"]})
{ acknowledged: true, insertedId: 'Books' }
project3>
```

3. Assume we model the records and relationships in Figure 1 using the Child-Referencing model (Slide 9 in MongoDB-3). Write a query to report the parent of “dbm”.

```
mongosh mongodb://127.0.0.

project3> db.question3part2.find({children:"dbm"})
[ { _id: 'Databases', children: [ 'MongoDB', 'dbm' ] } ]
project3>
```

4. Assume we model the records and relationships in Figure 1 using the Child-Referencing model (Slide 9 in MongoDB-3). Write a query to report the descendants of “Books”. The output should be an array containing values [“Programming”, “Languages”, “Databases”, “MongoDB”, “dbm”]

```
mongosh mongodb://127.0.0.

project3> var r = db.question3part2.findOne({_id: "Books"});

project3> var nt = [];

project3> var d= [];

project3> for (itr in r.children) {
...     nt.push(r.children[itr]);
... }
1
project3> while (nt.length > 0) {
...     var current = nt.shift();
...     d.push(current);
...     var itr = db.question3part2.findOne({_id: current});
...     for (i in itr.children) {
...         nt.push(itr.children[i])
...     }
... }
3
project3> d
[ 'Programming', 'Databases', 'Languages', 'MongoDB', 'dbm' ]
project3>
```