MONGODB

Reference Site

https://www.mikedane.com/databases/mongodb/
https://www.youtube.com/watch?v=DdvhZj7SsEM

Creating Collections:

```
use giraff;
db.createCollection("students");
db.students.drop();
```

Inserting

```
// Data Types
{
    string: "String of text",
    int: 405,
    double: 3.565,
    boolean: true, // false
    array: [1, 2, 3],
    object: {attr1: "attr1", attr2: "attr2"},
    date: new Date("<YYYY-mm-dd>"),
    object_id: <ObjectId>,
    no_value: null
}

/*
Additional Data Types
---------------------
Timestamp
Binary data
Regular expressions
JS Code
*/

// Inserting
db.students.insertOne({name: "Jack", major: "Biology", gpa: 3.5})
```

```
db.students.insertOne({name: "Claire", major: "Marketing", gpa: 3.7, awards: ["Valedictorian", "Summa Cum Laude"]} )
db.students.insertOne({name: "Evan", major: "Astronomy", gpa: 3.7, grades: [90, 88, 95, 78] } )
db.students.insertOne({name: "Kate", major: "Sociology", gpa: 3.2, contact: {phone: "333-3333", email: "student@school.edu"}})
db.students.insertOne({name: "Phil", major: "Chemistry", gpa: 2.5, startdate: new Date("2012-08-23")})
db.students.insertOne({_id: 4, name: "John", major: "Biology", gpa: 3.2})
db.students.insertMany([
    {name: "Mike", major: "Computer Science", gpa: 2.7},
    {name: "Andrea", major: "Math", gpa: 4.0, awards: ["Summa Cum Laude"]}
])
```

Find documents in collections

```
// Find all students
db.students.find( {} )

// Find the first 3 students
db.stuents.find( {} ).limit(3)

// Find all students and sort by name in ascending order
db.students.find( {} ).sort( {name: 1} )

// Find all students and sort by name in ascending order
db.students.find( {} ).sort( {gpa: -1, name: 1} )

// Find all biology majors
db.students.find( {major: "Biology"} )

// Find all student's with a phone number 333-3333
db.students.find( {contact: {phone: "333-3333", email: "student@school.edu"} } )

// Find all biology majors named Jack
db.students.find( {name: "Jack", major: "Biology"} )

// Final all students who are chemistry majors or named Jack
db.students.find( { $or: [ {name: "Jack"}, {major: "Chemistry"} ] } )

// Final all students with a gpa above 3.5
db.students.find( {gpa: {$gt: 3.5} } )
```

```
// Find all students with a gpa less than or equal to 3.2
db.students.find( {gpa: {$lte: 3.2} } ).sort({gpa: -1})                 // $eq, $ne, $lt, $lte, $gt, $gte

// Find all students with names in the array
db.students.find( {name: {$in: ["Kate", "Claire"]} } )   // $in, $nin

// Find all students who have awards
db.students.find( {awards: {$exists: true} } )           // false

// Find all db entries where the name is a string
// Type list - https://docs.mongodb.com/manual/reference/bson-types/
db.students.find({name: {$type: 2} })

// Find all students who's first grade is a 90
db.students.find( {"grades.0": 90 } )

// Find all students who have a grade greater than 80
db.students.find( {grades: {$elemMatch: { $gte: 80} } } )

// Find all students who have 4 grades recorded
db.students.find( {grades: {$size: 4 } } )
```

Updating, Replacing and Deleting

```
// same filters as inserting
db.stuents.updateOne(<filter>, <update>, <options>)

// Do this twice so we can change it back with updateMany
db.students.updateOne(
    {major: "Biology"},
    {
      $set:
        {major: "Bio"}
    }
)

db.students.updateMany(
    {major: "Bio"},
    {
      $set:
        {major: "Biology"}
```

```
    }
)

// replaceMany()
db.students.replaceOne(
    {major: "Bio"},
    {name: "new name", major: "new major", gpa: 4.0}
)

// Delete all documents
db.students.deleteMany({})

db.students.deleteOne({major: "Biology"})

db.students.deleteMany({gpa: {$gte: 3.5}})
```

BulkWrite

```
db.students.bulkWrite(
    [
      { insertOne :
        {
          "document" :
          {
            name: "Andrew", major: "Architecture", gpa: 3.2
          }
        }
      },
      { insertOne :
        {
          "document" :
          {
            name: "Terry", major: "Math", gpa: 3.8
          }
        }
      },
      { updateOne :
        {
          filter : { name : "Terry" },
          update : { $set : { gpa : 4.0 } }
        }
```

```
        },
        { deleteOne :
          { filter : { name : "Kate"} }
        },
        { replaceOne :
          {
            filter : { name : "Claire" },
            replacement : { name: "Genny", major: "Counsling", gpa: 2.4 }
          }
        }
    ],
    {ordered: false}
   );
```

Text Indexing

```
db.stores.insertMany(
  [
    { _id: 1, name: "Java Hut", description: "Coffee and cakes" },
    { _id: 2, name: "Burger Buns", description: "Gourmet hamburgers" },
    { _id: 3, name: "Coffee Shop", description: "Just coffee" },
    { _id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing" },
    { _id: 5, name: "Java Shopping", description: "Indonesian goods" }
  ]
)

db.stores.createIndex( { name: "text", description: "text" } )

db.stores.find({ $text: {$search: "Coffee" } })

db.stores.find({ $text: {$search: "Java Hut Coffee" } })

db.stores.find(
  { $text: { $search: "java hut coffee" } },
  { score: { $meta: "textScore" } }
).sort( { score: { $meta: "textScore" } } )
```

Aggregation

```
db.purchase_orders.insertMany(
    [
        {product: "toothbrush", total: 4.75, customer: "Mike"},
        {product: "guitar", total: 199.99, customer: "Tom"},
        {product: "milk", total: 11.33, customer: "Mike"},
        {product: "pizza", total: 8.50, customer: "Karen"},
        {product: "toothbrush", total: 4.75, customer: "Karen"},
        {product: "pizza", total: 4.75, customer: "Dave"}
        {product: "toothbrush", total: 4.75, customer: "Mike"},
    ]
)

// find out how many toothbrushes were sold
db.purchase_orders.count({product: "toothbrush"})

// Find list of all products sold
db.purchase_orders.distinct("product")

// Find the total amount of money spent by each customer
db.purchase_orders.aggregate(
    [
        {$match: {} },
        {$group: {_id: "$customer", total: { $sum: "$total"} } }
    ]
)

// Find how much has been spent on each product and sort it by price
db.purchase_orders.aggregate(
    [
        {$match: {} },
        {$group: {_id: "$product", total: { $sum: "$total"} } },
        {$sort: {total: -1}}
    ]
)

// Find how much money each customer has spent on toothbrushes and pizza
db.purchase_orders.aggregate(
    [
        {$match: {product: {$in: ["toothbrush", "pizza"]} } },
        {$group: {_id: "$product", total: { $sum: "$total"} } },
    ]
)
```

// https://docs.mongodb.com/manual/reference/operator/aggregation/
// Show the list of all pipeline operators

Schema validation

```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["name", "age", "hobbies", "address", "networth"],
      properties: {
        name: {
          bsonType: "string",
          description: "this should be of type string and is required",
        },
        age: {
          bsonType: "int",
          description: "this should be of type int and is required",
        },
        hobbies: {
          bsonType: "array",
          description: "this should be of type array and is required",
          items: {
            bsonType: "object",
            description: "this should be of object string and is required",
            required: ["title", "description"],
            properties: {
              title: {
                bsonType: "string",
                description: "this should be of type string and is required",
              },
              description: {
                bsonType: "string",
                description: "this should be of type string and is required",
              },
            },
          },
        },
        address: {
          bsonType: "object",
```

```
        description: "this should be of type object and is required",
        required: ["city", "street"],
        properties: {
          city: {
            bsonType: "string",
            description: "this should be of type string and is required",
          },
          street: {
            bsonType: "string",
            description: "this should be of type string and is required",
          },
        },
      },
      networth: {
        bsonType: "long",
        description: "this should be of type long and is required",
      },
    },
  },
 },
});
```