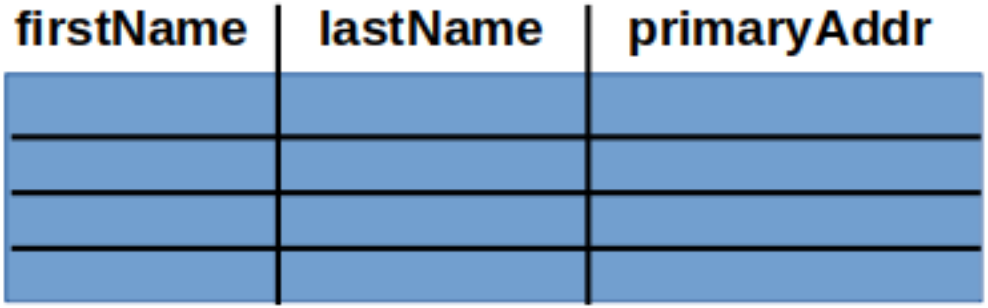
Intro to MongoDB

CMPSC 305 – Database Systems

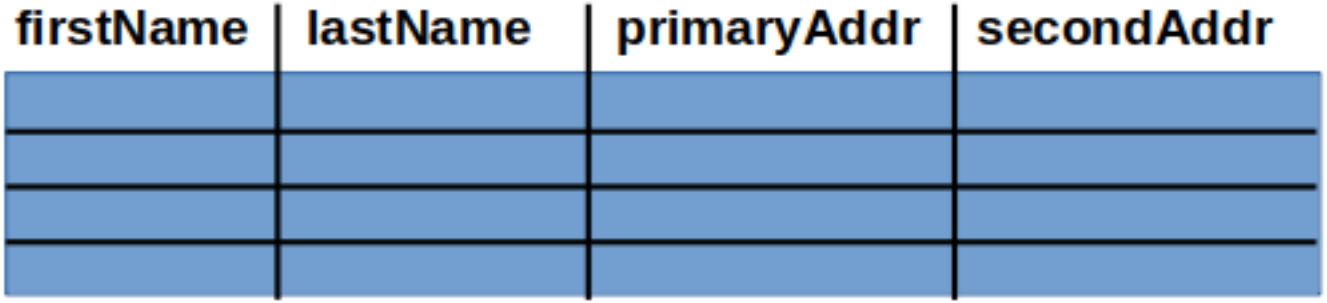
ALLEGHENY COLLEGE

The Problem with SQL



- Let's say that we have a (perfectly) working SQL table
- The schema has been designed and coded for current data requirements

Table Update (i)



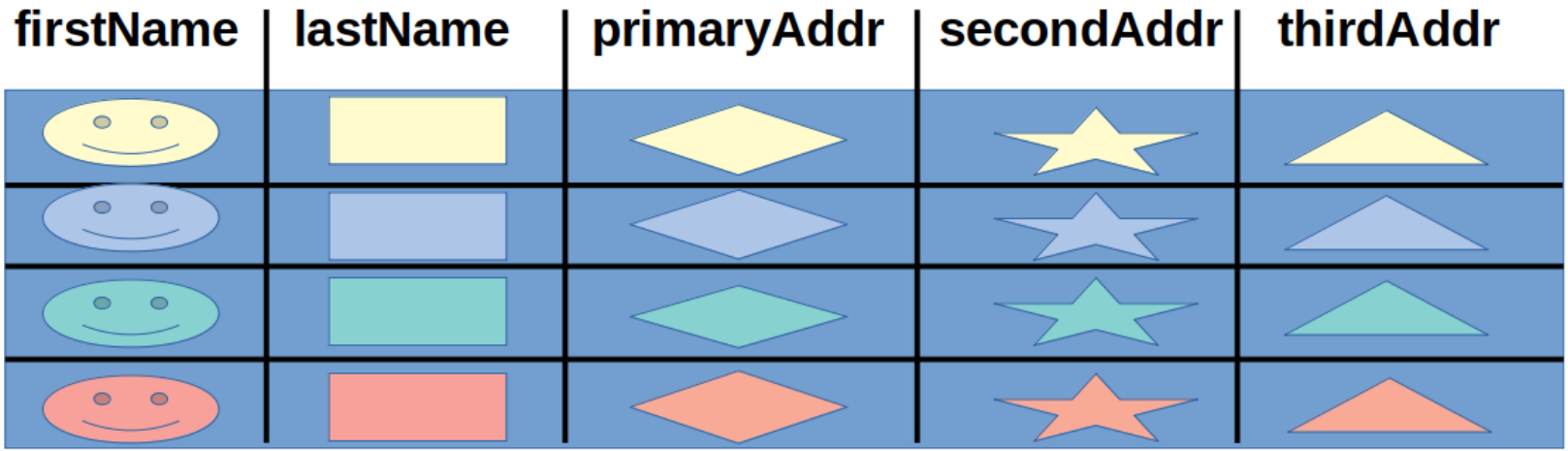
- The data we collect has changed.
- We need to update our schema for the new data requirements

Table Update (ii)

firstName	lastName	primaryAddr	secondAddr	thirdAddr

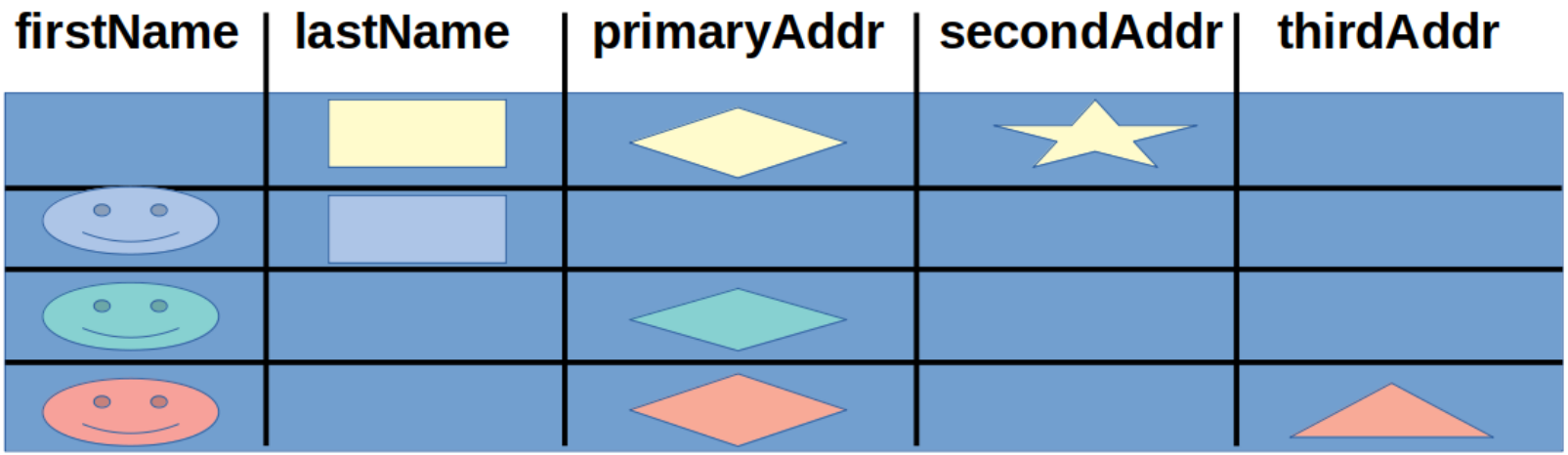
- Our needs have changed again, and the SQL table must be updated.
- The schema is reprogrammed

Expectations



- We expect that the table will be full when in use
- Expectations are not always fulfilled…

In Reality, Much Data is Missing



- But, in reality, much of the table is empty!
- The table can easily get huge and be hard to manage.

We Might Ask Ourselves…

- What can we do to stop having to redesign our database schema with our changing data?
- Is SQL the right type of database management system for our changing data requirements?

A NoSQL Database Management System

- NoSQL: Not Only SQL database systems that support SQL-like query languages, but are used increasingly in big data applications and real-time web applications.
- The stored data is allowed to change

- https://www.mongodb.com/

Philosophy of MongoDB

Non-relational DB

- Document Identifiers (id) will be created for each document, field name reserved by system
- Application tracks the schema and mapping
- Uses JSON, BSON (B for binary inputs)
- Written in C++
- Supports APIs (drivers) in many computer languages

  JavaScript, Python, Golang, Ruby, Perl, Java, Java Scala, C#, C++, Haskell, Erlang

Database Language Guide

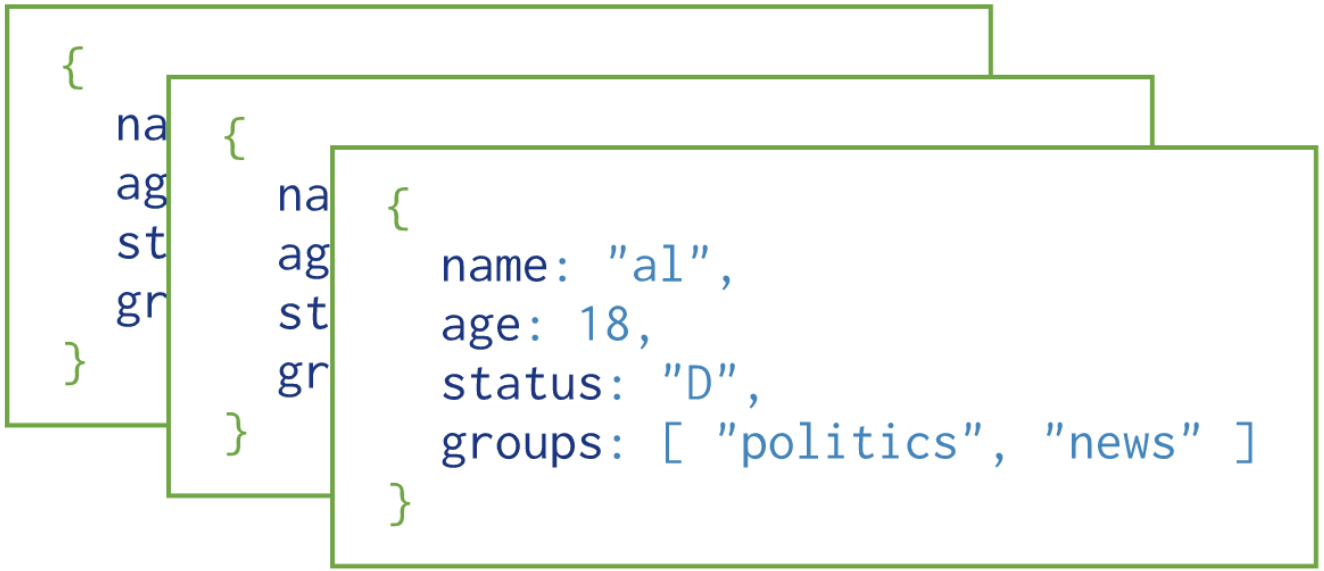
SQL systems versus NoSQL

RDBMS		MongoDB
Database	⇨	Database
Table, View	⇨	Collection
Row	⇨	Document (BSON)
Column	⇨	Field
Index	⇨	Index
Join	⇨	Embedded Document
Foreign Key	⇨	Reference
Partition	⇨	Shard

- The terms are different, but their meanings are similar
- Schema-less, collections (like tables) are populated by any data
- Documents are similar to the tuples of Sqlite3 programming

Database Language Guide

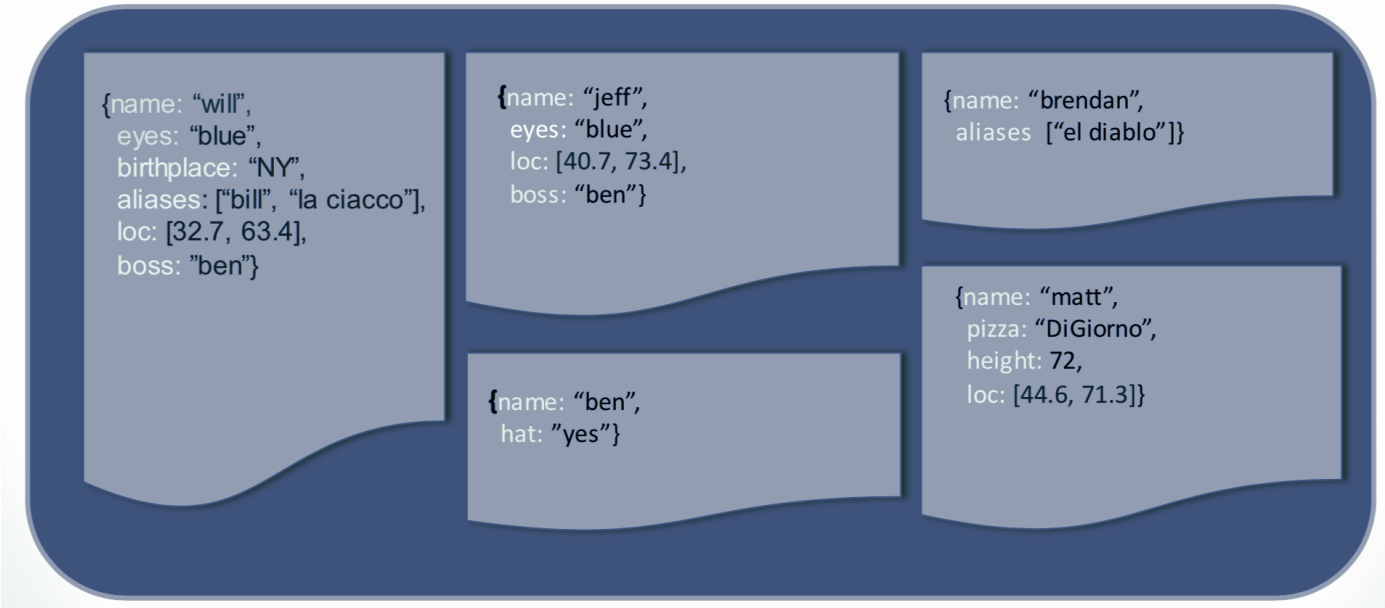
SQL systems versus NoSQL



- No pre-defined data schema
  - Data may be entered in absence of a defined schema
- Documents (rows) of collections (DB's) may have different types of data

Schema Free

Mostly similar documents



- Sometimes not all the data is available to create a document.
- The query interprets missing data as NULL entries

Styles of Storing Data

Relational Database

Student_Id	Student_Name	Age	College
1001	Chaitanya	30	Beginnersbook
1002	Steve	29	Beginnersbook
1003	Negan	28	Beginnersbook

MongoDB

```
{
 "_id": ObjectId("....."),
 "Student_Id": 1001,
 "Student_Name": "Chaitanya",
 "Age": 30,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1002,
 "Student_Name": "Steve",
 "Age": 29,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1003,
 "Student_Name": "Negan",
 "Age": 28,
 "College": "Beginnersbook"
}
```

JSON and MongoDB Code

- **Data is in name / value pairs**
- A name/value pair consists of a field name followed by a colon, followed by a value:

  Example: { "name": "R2-D2 "}
- **Data is separated by commas**

  Example: { "name": "R2-D2", race : "Droid"}
- **Curly braces hold objects**

  Example: {"name": "R2-D2", race : "Droid",
  affiliation: "rebels"}
- **An array is stored in brackets []**

  Example [{ "name": "R2-D2", race : "Droid",
  affiliation: "rebels"},
  { "name": "Yoda", affiliation: "rebels"}]

CRUD Operations

Create, Read, Update and Delete

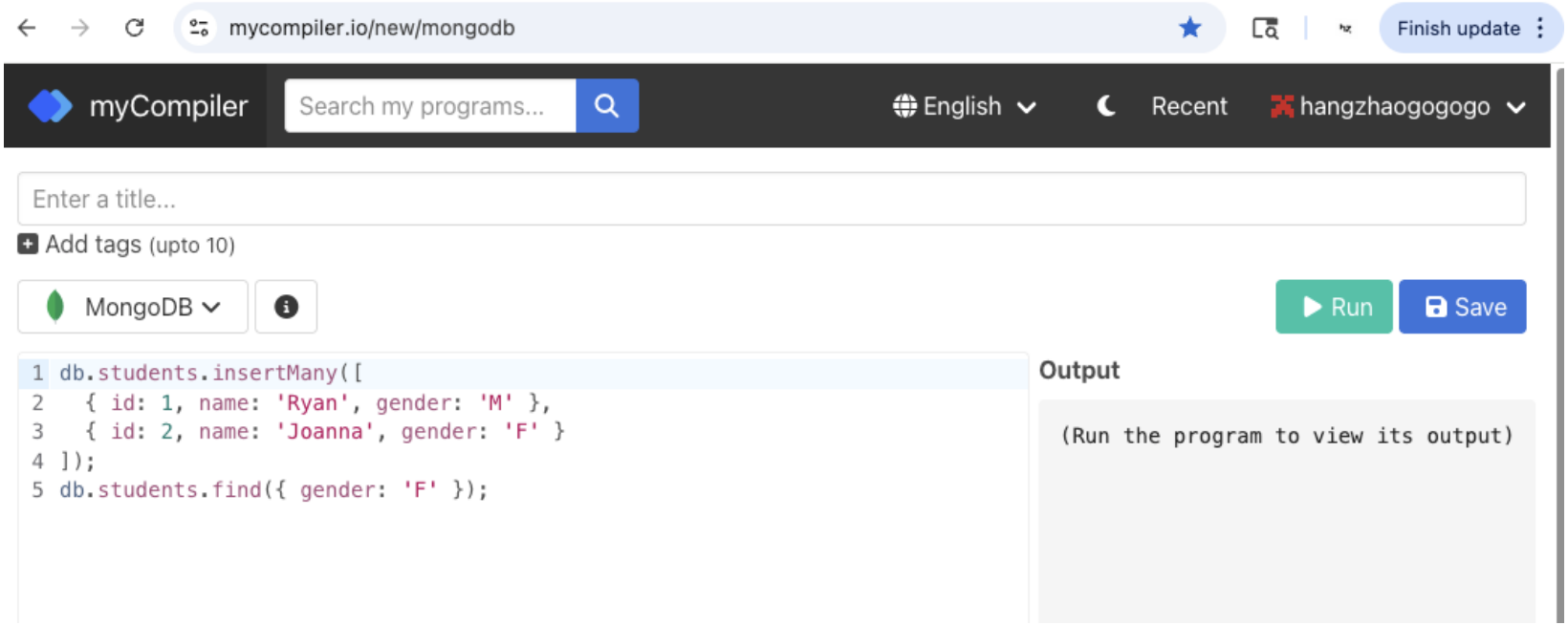
- **Db.collection** specifies the collection or the table in which to store the document (tuple)
- **Create**
  - db.collection.insert()
  - db.collection.save()
  - db.collection.update()
- **Read**
  - db.collection.find()
  - db.collection.findOne()
- **Update**
  - db.collection.update()
- **Delete**
  - db.collection.remove()

Let’s code

Online coding to test the new DB

Let's code

Online MongoDB



- https://www.mycompiler.io/new/mongodb

Let's code

output

Output

```
 mycompiler_mongodb> ... ... ... {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6906c2e5e9b991bbe56b128c'),
    '1': ObjectId('6906c2e5e9b991bbe56b128d')
  }
}
mycompiler_mongodb> [
  {
    _id: ObjectId('6906c2e5e9b991bbe56b128d'),
    id: 2,
    name: 'Joanna',
    gender: 'F'
  }
]
mycompiler_mongodb>

[Execution complete with exit code 0]
```

Simple Collections

Enter data as JSON code

Insert each document individually into the Furniture collection

```
db.Furniture.drop()
        db.Furniture.insertOne({chair:"wood"})
        db.Furniture.insertOne({chair:"metal"})
        db.Furniture.insertOne({chair:"plastic"})
        db.Furniture.insertOne({table:"glass"})
        db.Furniture.insertOne({table:"wood"})
        db.Furniture.insertOne({table:"metal"})
        db.Furniture.insertOne({lamp:"brass"})
        db.Furniture.insertOne({lamp:"glass"})
        db.Furniture.insertOne({lamp:"silver"})
```

db.Furniture.find({searchSpace},{showAttrib:1})

Find everything: db.Furniture.find({},{})

Let's code

output


```
1  db.Furniture.drop()
2   db.Furniture.insertOne({chair:"wood"})
3   db.Furniture.insertOne({chair:"metal"})
4   db.Furniture.insertOne({chair:"plastic"})
5   db.Furniture.insertOne({table:"glass"})
6   db.Furniture.insertOne({table:"wood"})
7   db.Furniture.insertOne({table:"metal"})
8   db.Furniture.insertOne({lamp:"brass"})
9   db.Furniture.insertOne({lamp:"glass"})
10  db.Furniture.insertOne({lamp:"silver"})
11
12 db.Furniture.find({},{lamp:1})
13
```

Find all lamps: db.Furniture.find({},{lamp:1})

Let's code

output

```
mycompiler_mongodb>
mycompiler_mongodb> [
  { _id: ObjectId('6906c3d68ba14fc50e6b128c') },
  { _id: ObjectId('6906c3d68ba14fc50e6b128d') },
  { _id: ObjectId('6906c3d78ba14fc50e6b128e') },
  { _id: ObjectId('6906c3d78ba14fc50e6b128f') },
  { _id: ObjectId('6906c3d78ba14fc50e6b1290') },
  { _id: ObjectId('6906c3d78ba14fc50e6b1291') },
  { _id: ObjectId('6906c3d78ba14fc50e6b1292'), lamp: 'brass' },
  { _id: ObjectId('6906c3d78ba14fc50e6b1293'), lamp: 'glass' },
  { _id: ObjectId('6906c3d78ba14fc50e6b1294'), lamp: 'silver' }
]
mycompiler_mongodb>

[Execution complete with exit code 0]
```

Find all lamps: db.Furniture.find({},{lamp:1})

Let's code

output

MongoDB

```
1  db.Furniture.drop()
2   db.Furniture.insertOne({chair:"wood"})
3   db.Furniture.insertOne({chair:"metal"})
4   db.Furniture.insertOne({chair:"plastic"})
5   db.Furniture.insertOne({table:"glass"})
6   db.Furniture.insertOne({table:"wood"})
7   db.Furniture.insertOne({table:"metal"})
8   db.Furniture.insertOne({lamp:"brass"})
9   db.Furniture.insertOne({lamp:"glass"})
10  db.Furniture.insertOne({lamp:"silver"})
11
12 db.Furniture.find({},{lamp:1}).pretty()
13
```

Find all lamps, use formatting: db.Furniture.find({},{lamp:1}).pretty()

Let's code

output

```
mycompiler_mongodb>
mycompiler_mongodb> [
  { _id: ObjectId('6906c42393cfa764426b128c') },
  { _id: ObjectId('6906c42493cfa764426b128d') },
  { _id: ObjectId('6906c42493cfa764426b128e') },
  { _id: ObjectId('6906c42493cfa764426b128f') },
  { _id: ObjectId('6906c42493cfa764426b1290') },
  { _id: ObjectId('6906c42493cfa764426b1291') },
  { _id: ObjectId('6906c42493cfa764426b1292'), lamp: 'brass' },
  { _id: ObjectId('6906c42493cfa764426b1293'), lamp: 'glass' },
  { _id: ObjectId('6906c42493cfa764426b1294'), lamp: 'silver' }
]
mycompiler_mongodb>

[Execution complete with exit code 0]
```

Find all lamps, use formatting: db.Furniture.find({},{lamp:1}).pretty()

Simple Collections

Simple Example of Queries

Query all documents in the Furniture collection

```
db.Furniture.find({},{})
```

Query all Lamp types across all collections

```
db.Furniture.find({},{lamp:1})
db.Furniture.find({},{lamp:1,_id:0})
```

Simple Collections

Simple Example of Queries

Query Lamp types from the Furniture collection

```
// SELECT lamp FROM Furniture WHERE lamp == ``brass'';
db.Furniture.find({lamp:"brass"})
db.Furniture.find({lamp:"glass"})
db.Furniture.find({lamp:"silver"})

// do not show object id's
db.Furniture.find({lamp:"silver"},{_id:0})
```

Insert many documents into the Inventory collection

Inserting

```
db.inventory.insertMany([
{ item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
{ item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
{ item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
{ item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
{ item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Insert many documents into the Inventory collection

SELECT * FROM inventory

```
db.inventory.find( {},{} )
db.inventory.find( {},{} ).pretty()
```

SELECT item FROM inventory

```
db.inventory.find({},{"item":1}).pretty()
```

SELECT * FROM inventory WHERE item == “postcard”

```
db.inventory.find({"item":"postcard"},{})
db.inventory.find({"item":"postcard"},{}).pretty()
```

Queries from the Inventory collection

SELECT * FROM inventory WHERE status = ”D”

```
db.inventory.find( { status: "D" } )
```

SELECT * FROM inventory WHERE status in (”A”, ”D”)

```
db.inventory.find({status:{ $in: [ "A", "D" ]}})
```

SELECT * FROM inventory WHERE status == “D”)

```
db.inventory.find({ status: "D" },{})
```

Show me where the size = “h” and size = 10)

```
db.inventory.find( {"size.h":10} ).pretty()
```

See more on this at https://www.mongodb.com/docs/manual/tutorial/query-documents/

Consider this!

- Can you go back to the above examples to query other fascinating information?
- Can you create and populate a new Mongo database?
- Can you write sophisticated queries in your database?