```
const fs = require('fs');

let items = JSON.parse(fs.readFileSync('./mongo-items.json','utf8'))
let set = new Set();
items.forEach(item=>
    {
        Object.keys(item).forEach(key=>set.add(key))
    })

//Given the items database, with these fields.
console.log(set)
//=> { '_id', 'name', 'description', 'category', 'stackSize' }
```

Note: Table and Collection will be synonims in this cheat sheet.

Get all data

```
SELECT * FROM items;
```

```
db.items.find().pretty();
//.pretty() is used for a readable json format
```

Get specific fields from a database

```
SELECT name,description FROM items;
```

```
db.article.aggregate
(
  { $project : { name : 1, description : 1 } }
);
```

However you can also do it with the find method

```
db.items.find({},{name:1,description:1})
```

Queries with constraints

We will find that just selecting everything from a database is pretty much something that realisticly we won't do. The things that we want from a database are more specific, each person might ask for different things, thus the need to add constraints.

SQL

```
SELECT name,stackSize FROM items
WHERE stackSize <= 20;
```

MongoDB

```
db.items.find(
    {stackSize:{$lte:20}}
)
```

We have different sets of operators here's a list

Logical Operators / Comparison Operators

- Docs Manual Here'a list of the operators.
- Equal To
- Not Equal To

- Lesser Than
- Lesser Than or Equal
- Greater Than
- Greater Than or Equal
- IN => [2,5,10,18]
- NOT IN => [30,4,9,81]
- Existence / NULL / Undefined checks

Equal [=]

If we want to EXACTLY look for something, we will use equals.

```
SELECT * FROM items
WHERE stackSize = 1;
```

However in Mongo there's no equal operator per se, we simple query the data with the json notation.

```
db.items.find({stackSize:1})
```

Alternatively there's an `$eq` operator. You can use a regular expression here.

- Docs Manual

Not Equal

If we want to do the opposite, we will use the not equals operator. For SQL is `!=` or `<>`

```
SELECT * FROM items
WHERE stackSize != 1;
```

For MongoDB we use the `$ne` which stands for `Not Equal` or `Inequailty`.

```
db.items.find({stackSize:{$ne:1}})
```

- Docs Manual

Note: A big difference between SQL and Mongo here is that SQL will not return rows where the data doesn't exist, and mongo will return them even if they are undefined.

Does the data exists? (NULL)

I believe it's a good moment, even if we are just getting started with operators to talk about `NULL` or `inexistence`. Because sometimes, for random reasons your data won't have all fields with data. In our example, some itemes don't have a `stackSize`, yet they are still items.

```
SELECT * FROM items
WHERE stackSize IS NULL;
```

Similar to the `equals` and `not equals` operator. You have a `NOT` after the `IS` keyword.

```
SELECT * FROM items
WHERE stackSize IS NOT NULL;
```

For Mongo we can use the `$exists` operator.

```
//Exists = True is equivalent to IS NOT NULL
db.items.find({stackSize:{$exists:true}})
//Exists = True is equivalent to IS NULL
db.items.find({stackSize:{$exists:false}})
```

You can also use the `$ne` or `$eq` operator to check for `null`.

```
db.items.find({stackSize:{$eq:null}})
```

The following four operators work pretty much the same, but they inquire for different things. As their name suggest you wanna now if something is lesser or greater than other things.

Lesser than [<]

```
SELECT * FROM items
WHERE stackSize < 10;
```

Mongo Operator: `$lt`

```
db.items.find({stackSize:{$lt:10}})
```

Lesser than or equal [<=]

```
SELECT * FROM items
WHERE stackSize < 10;
```

Mongo Operator: `$lte`

```
db.items.find({stackSize:{$lte:10}})
```

Greater than [>]

```
SELECT * FROM items
WHERE stackSize < 10;
```

Mongo Operator: `$gt`

```
db.items.find({stackSize:{$gt:10}})
```

Greater than or equal [>=]

```
SELECT * FROM items
WHERE stackSize < 10;
```

Mongo Operator: `$gte`

```
db.items.find({stackSize:{$gte:10}})
```

IN - is your data inside this list of things?

The `IN` operator pretty much asks for a lists of things. Is your data inside this array?

```
SELECT * FROM items
WHERE stackSize IN (10,20,30,40,50);
```

```
db.items.find({stackSize:{$in:[10,20,30,40,50]}})
```

Just like Equal/NotEqual, Null/Exists, you can ask for `NOT IN`.

```
SELECT * FROM items
WHERE stackSize NOT IN (5,10,15,20);
```

```
db.items.find({stackSize:{$nin:[5,10,15,20]}})
```

AND Operator - AND

Just like in programming, we will use the `AND` operator. This is used when you want 2 or more conditions to be met. Example: Find an Item that is from the `Usable` category and that it has a `stackSize` greater than 20. If `ALL` conditions are met, it will return a data set for the conditions. There's no need to check for other conditions if one is NOT met.

```
SELECT * FROM items
WHERE category="Usable" AND stackSize > 20;
```

For Mongo it will look a bit different, since with an `AND` its plural for 2 or more, we will use an array for this operator.

```
db.items.find( { $and: [ { category: "Usable" }, {stackSize:{$gt:20}} ] } )
```

As you can see, the `$and` relies on an array to fulfill as many conditions as desired.

OR Operator - OR

Just like `AND` it will prompt for an array or conditions. However the difference is that `OR` will trigger if `ANY` of the conditions are met. There's no need to check for other conditions if one is met.

```
SELECT * FROM items
WHERE category="Usable" OR stackSize > 20;
```

For Mongo it will look a bit different, since with an `AND` its plural for 2 or more, we will use an array for this operator.

```
db.items.find( { $or: [ { category: "Usable" }, {stackSize:{$gt:20}} ] } )
```

Between / Not Between

The between method doesn't exist on Mongo, however, we know that the "primitive" way to do a between functionality is to put things between a greater/lesser than equal conditions. This is, if we want to find an Item that stackSize is between 10 AND 30, we can say.

```
SELECT * FROM items
WHERE stackSize >= 10 AND stackSize <=30;
```

In SQL we can use a `BETWEEN` keyword.

```
SELECT * FROM items
WHERE stackSize BETWEEN 10 AND 30;
```

For mongo we will need the logical operators.

```
db.items.find({stackSize: {$gte:10,$lte:30}} )
```

There's also the `NOT BETWEEN`, which is basically a negation of the range that we wanted.

```
SELECT * FROM items
WHERE stackSize NOT BETWEEN 10 AND 30;
```

```
SELECT * FROM items
WHERE NOT (stackSize >= 10 AND stackSize <= 30);
```

```
db.items.find({ stackSize:{$not:{$gte:10,$lte:30}}} )
```

NOT Operator

Like the previous example, we are asking for the opposite of what we asking. I want these items that `DO NOT` fulfill this conditions.

- Docs Manual

```
SELECT * FROM items
WHERE NOT (stackSize >= 10 AND stackSize <= 30);
```

```
db.items.find({ stackSize:{$not:{$gte:10,$lte:30}}} )
```