Introduction to NoSQL

Introduction to NoSQL

- Background
  - As data storage costs rapidly decreased…
    - The amount of **data and query increased**
    - Data came in **various shapes and sizes**—structured, semistructured, and polymorphic—and **defining the schema in advance became nearly impossible**
  - **NoSQL databases** allow…
    - Developers to store huge amounts of unstructured data, giving them a lot of flexibility
  - Emerged in the late 2000s
    - **Amazon DynamoDB**: Hosted and scalable database service by Amazon with the data stored in Amazons cloud (DeCandia et al. 2007)
    - **Google Bigtable**: Google's NoSQL Big Data database service(Search, Analytics, Maps, and Gmail) (Chang et al. 2008)

Introduction to NoSQL

- Concept
  - NoSQL databases (aka "not only SQL") are **non tabular**, and store data differently than relational tables

- Characteristics
  - Allows **various types of data** to be nested **within a single data structure**
  - Has a structure in which data is **dispersely stored and processed** by connecting dozens of general servers
  - Has a **flexible schema**

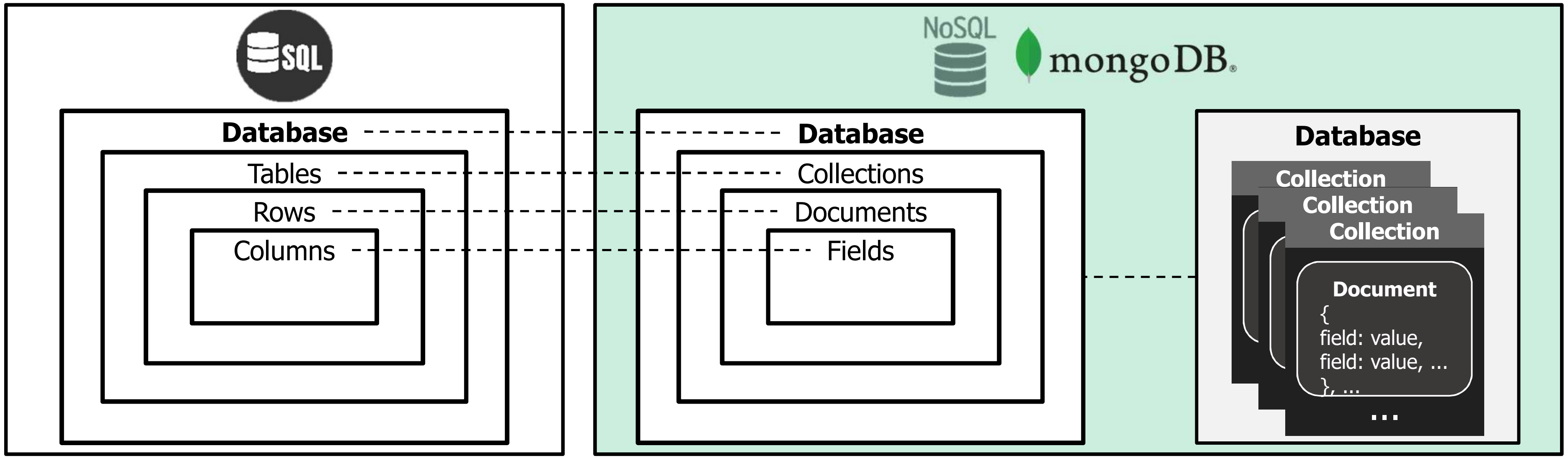
Introduction to NoSQL

- NoSQL DB Types

DB Type	Data type	Concept	Example
Document databases	JavaScript Object Notation(JSON) objects	▪ Each document contains **pairs of fields and values** ▪ The values can typically be **a variety of types** including things like strings, numbers, booleans, arrays, or objects ▪ Each document can **horizontally scale-out** to accomodate large data volumes	MongoDB
Key-value databases	Keys and values	▪ Each item contains **keys and values** ▪ A value can typically only be retrieved by referencing its key, so learning **how to query for a specific key-value pair is typically simple**	DynamoDB, Redis
Wide-column stores	Tables, rows, and dynamic columns	▪ Each item contains data in **tables, rows, and dynamic columns** ▪ A dynamic column provide **a lot of flexibility over relational databases** because **each row is not required to have the same columns**	Cassandra, Hbase
Graph databases	Nodes and edges	▪ Each item contains **data in nodes and edges** ▪ **Nodes** typically store **information about people, places, and things** while **edges** store **information about the relationships between the nodes**	Neo4j, JanusGraph

SQL vs. NoSQL

DB	Feature	Application
SQL	▪ Static schema ▪ Need to clearly define relationships between entities ▪ Ensure data integrity/consistency ▪ Avoid data duplication	▪ When data requirements are clear ▪ When data structure changes infrequently
NoSQL	▪ Dynamic schema ▪ No need to define relationship between entities → Any relationship can be defined ▪ Excellent scalability and scalability → All types of data can be stored	▪ When data requirements are ambiguos ▪ When data structure changes frequently → When data needs to be horizontally scaled ▪ When processing large amounts of data



SQL vs. NoSQL

SQL

id	name	item
1	Charlie	4
2	David	5
3	Ellis	6

name_id	name
1	Charlie
2	David
3	Ellis

item_id	item
1	4
2	5
3	6
4	**7**

name_id	item_id	item
1	1	4
2	2	5
3	3	6
3	**4**	**7**

NoSQL

{id: 1, name: ‘Charlie’, item: 4}

{id: 2, name: ‘David‛, item: 5}

{id: 3, name:: ‘Ellis’, item: 6}

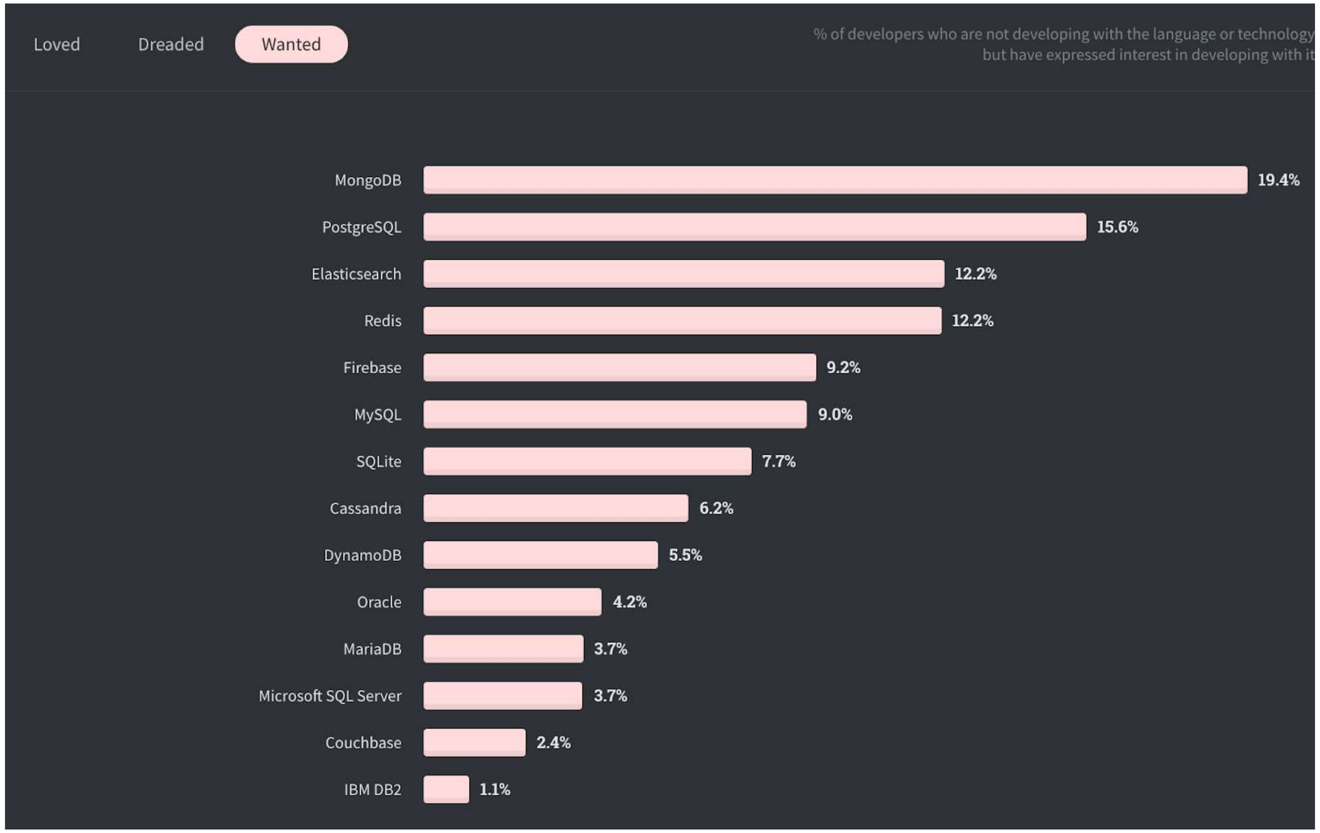
{id: 1, name: ‘Charlie’, item: 4}

{id: 2, name: ‘David‛, item: 5}

{id: 3, name:: ‘Ellis’, item: **[6,7]**}

Introduction to MongoDB

- Why MongoDB?
  - Most wanted NoSQL DBMS in the world

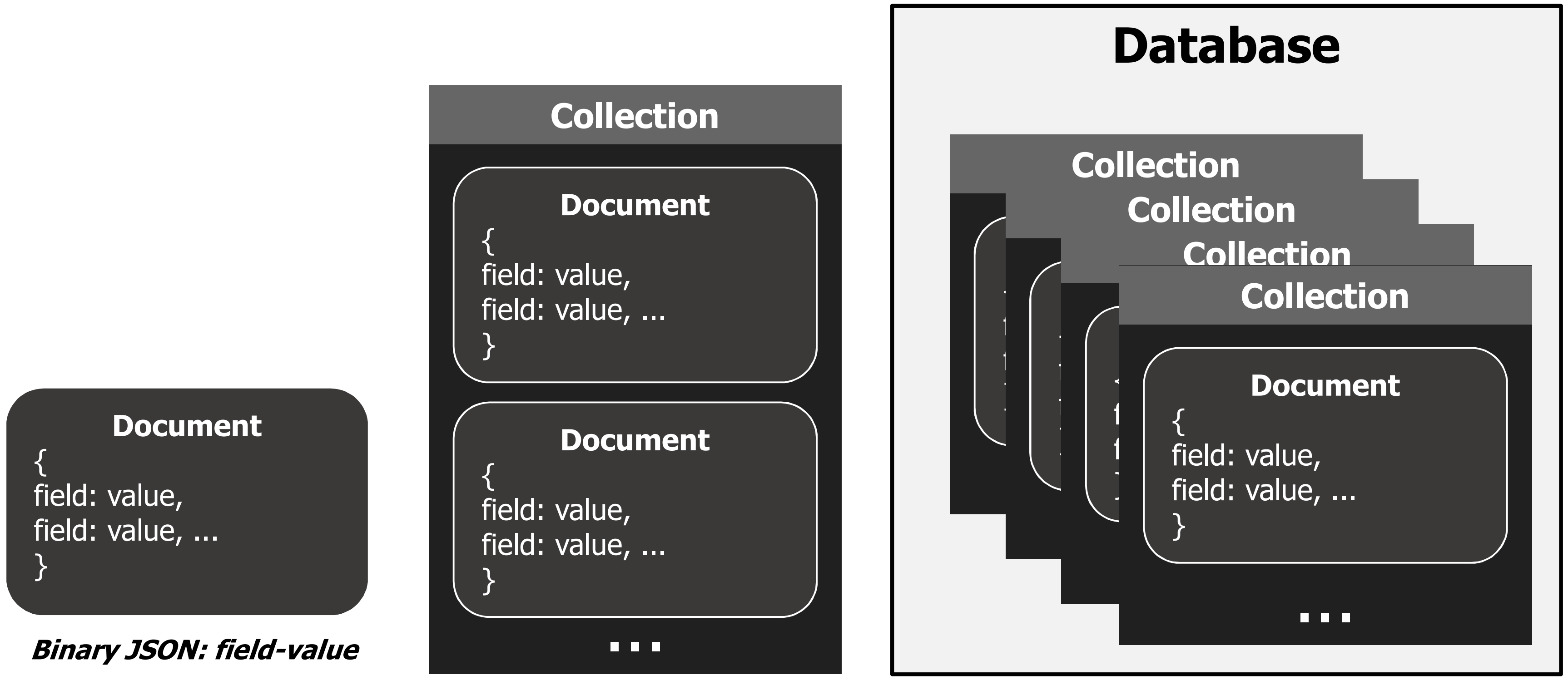


Database	Wanted
MongoDB	19.4%
PostgreSQL	15.6%
Elasticsearch	12.2%
Redis	12.2%
Firebase	9.2%
MySQL	9.0%
SQLite	7.7%
Cassandra	6.2%
DynamoDB	5.5%
Oracle	4.2%
MariaDB	3.7%
Microsoft SQL Server	3.7%
Couchbase	2.4%
IBM DB2	1.1%

Image courtesy of Stack Overflow 2020 Developer Survey, Most Wanted Databases

The Structure of MongoDB

- Document – Collection – Database



Data Type of MongoDB

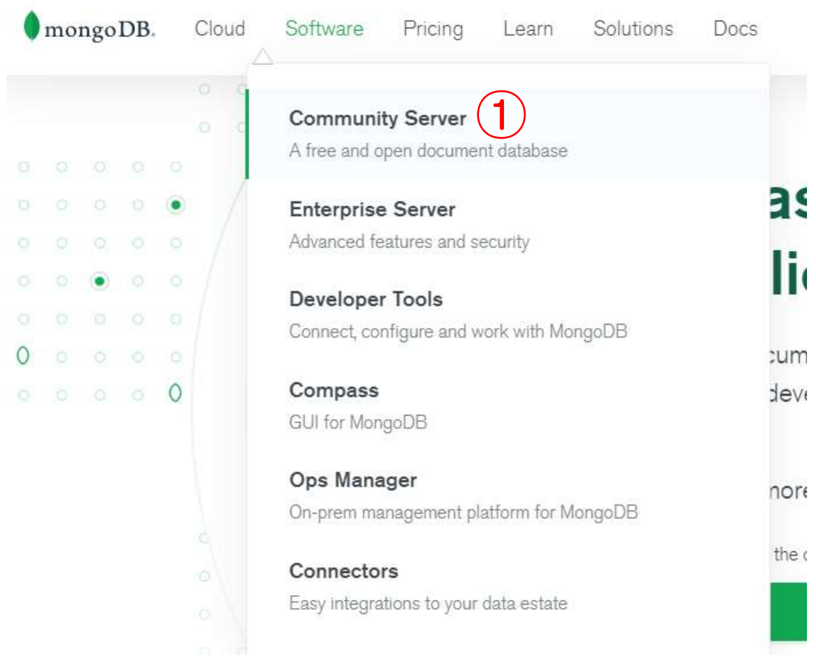
Type	Description	Example
Date	This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.	2021-05-12T01:30:11.000+00:00
Null	This type is used to store a Null value.	null
Integer	This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.	1, 100, 1000
Double	This type is used to store floating point values.	1.123, 3.1415
String	This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.	"hello", "world"
Object	This datatype is used for embedded documents.	{filed1: 'value1', field2: 'value2'}
Boolean	This type is used to store a boolean (true/ false) value.	true, false
Arrays	This type is used to store arrays or list or multiple values into one key.	{1, 3.14, null, {x:1}, true}

Practices

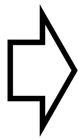
- Objectives
  - Setup MongoDB
  - Look around MongoDB shell
    - CRUD: Create, Read, Update, Delete
  - Query operator

Practices

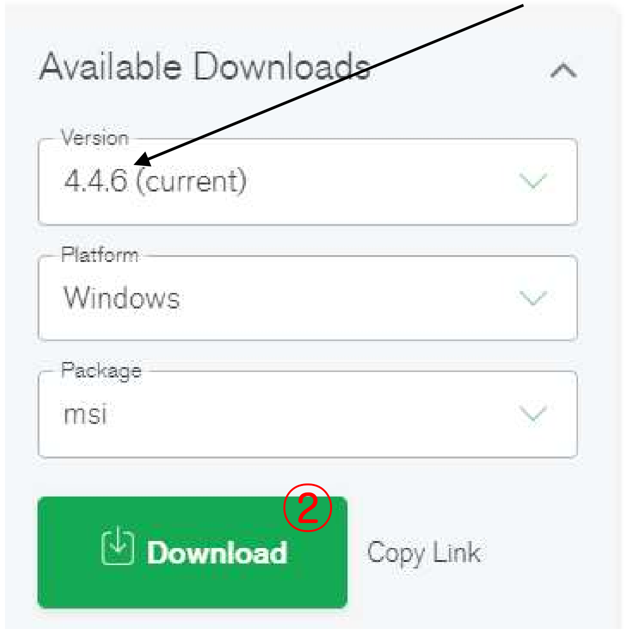
- Setup MongoDB
  - OS: Windows 10
  - https://www.mongodb.com/



mongoDB.
Cloud
Software
Pricing
Learn
Solutions
Docs
Community Server ①
A free and open document database
Enterprise Server
Advanced features and security
Developer Tools
Connect, configure and work with MongoDB
Compass
GUI for MongoDB
Ops Manager
On-prem management platform for MongoDB
Connectors
Easy integrations to your data estate



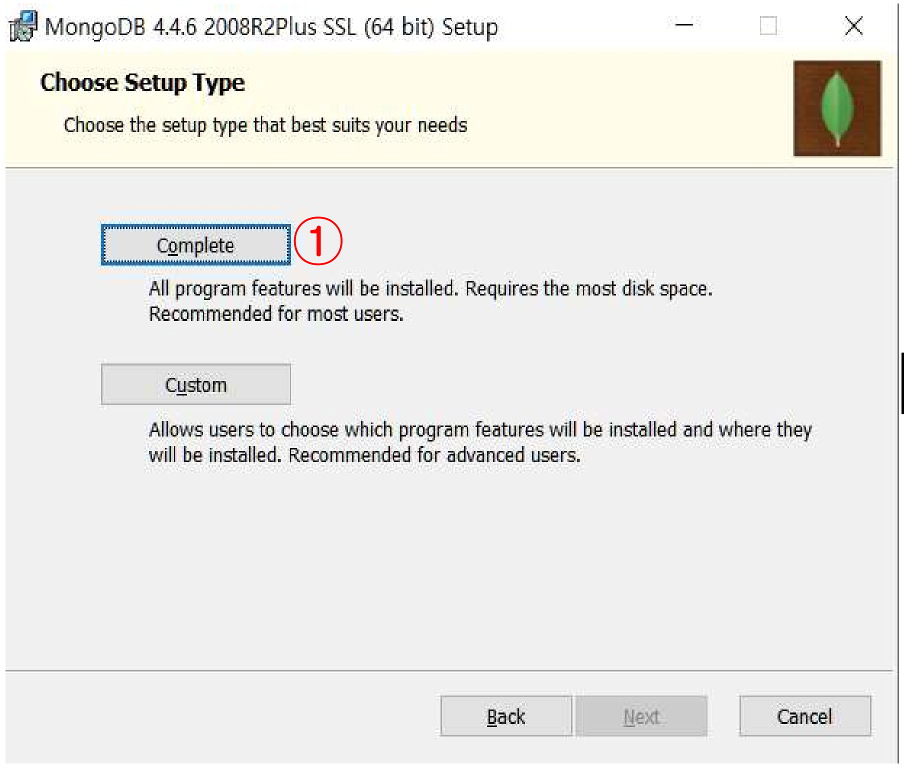
even number: stabilized version
odd number: under developing



Version: 4.4.6 (current)
Platform: Windows
Package: msi

Practices

- Setup MongoDB
  - Next, next …



MongoDB 4.4.6 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

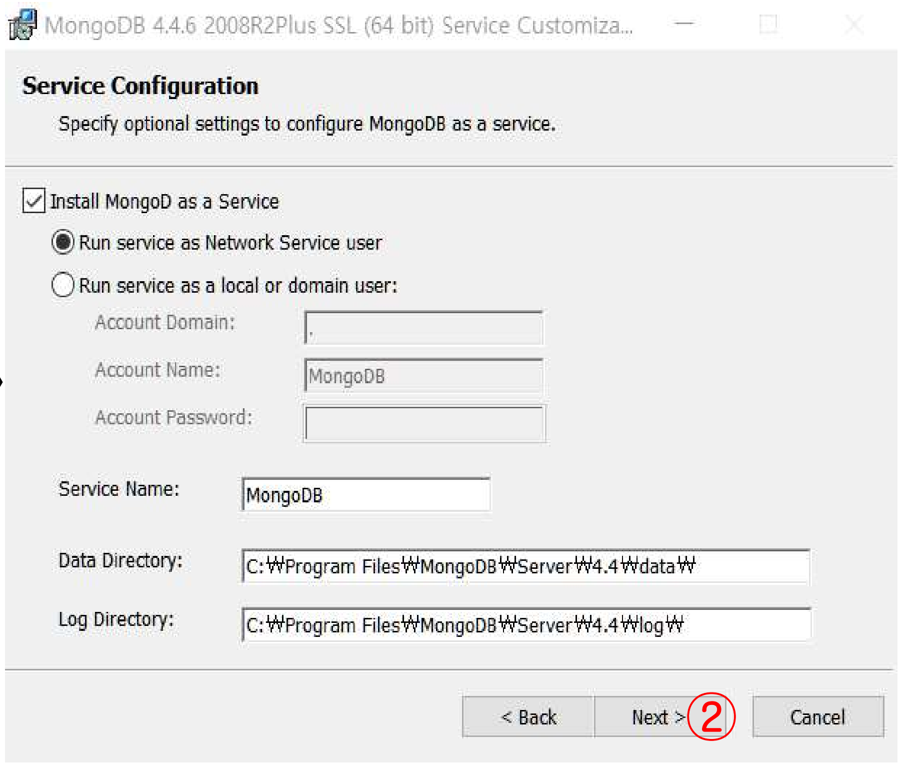
Choose the setup type that best suits your needs

Complete ①

All program features will be installed. Requires the most disk space. Recommended for most users.

Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.



MongoDB 4.4.6 2008R2Plus SSL (64 bit) Service Customiza...

Service Configuration

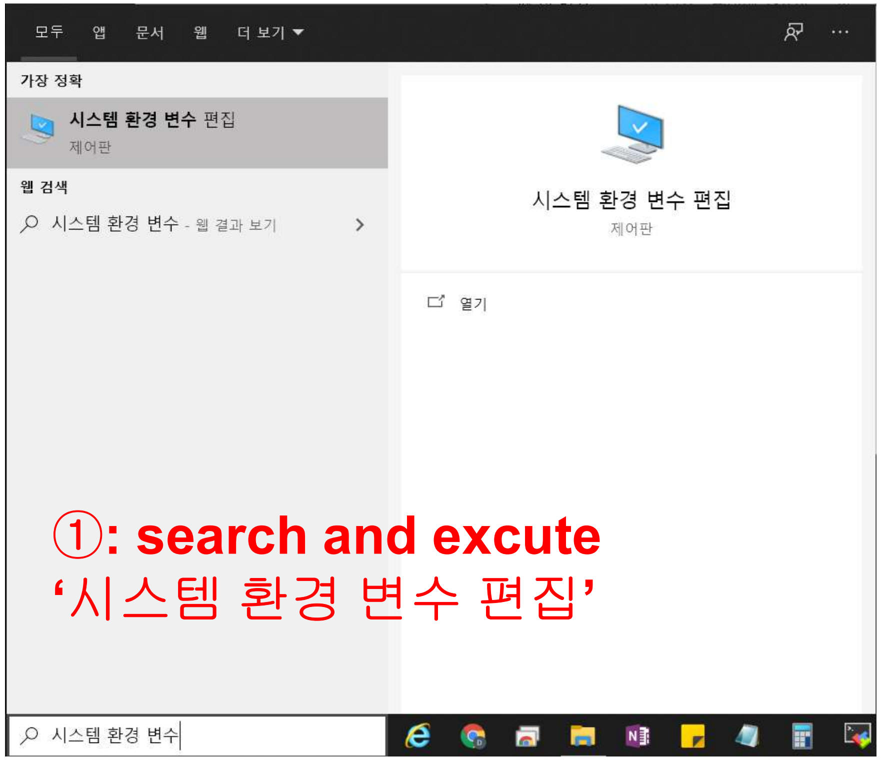
Specify optional settings to configure MongoDB as a service.

- [✓] Install MongoD as a Service
- ● Run service as Network Service user
- ○ Run service as a local or domain user:

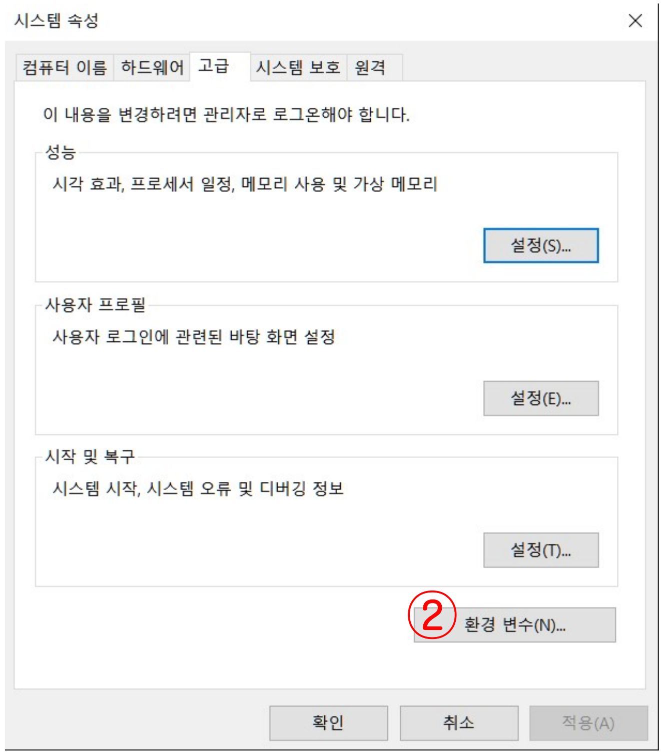
Account Domain:	.
Account Name:	MongoDB
Account Password:	
Service Name:	MongoDB
Data Directory:	C:₩Program Files₩MongoDB₩Server₩4.4₩data₩
Log Directory:	C:₩Program Files₩MongoDB₩Server₩4.4₩log₩

Practices

- Setup MongoDB
  - Please check and remember your mongoDB path
    e.g., *C:₩Program Files₩MongoDB₩Server₩4.4₩bin*
  - Set the environment variables

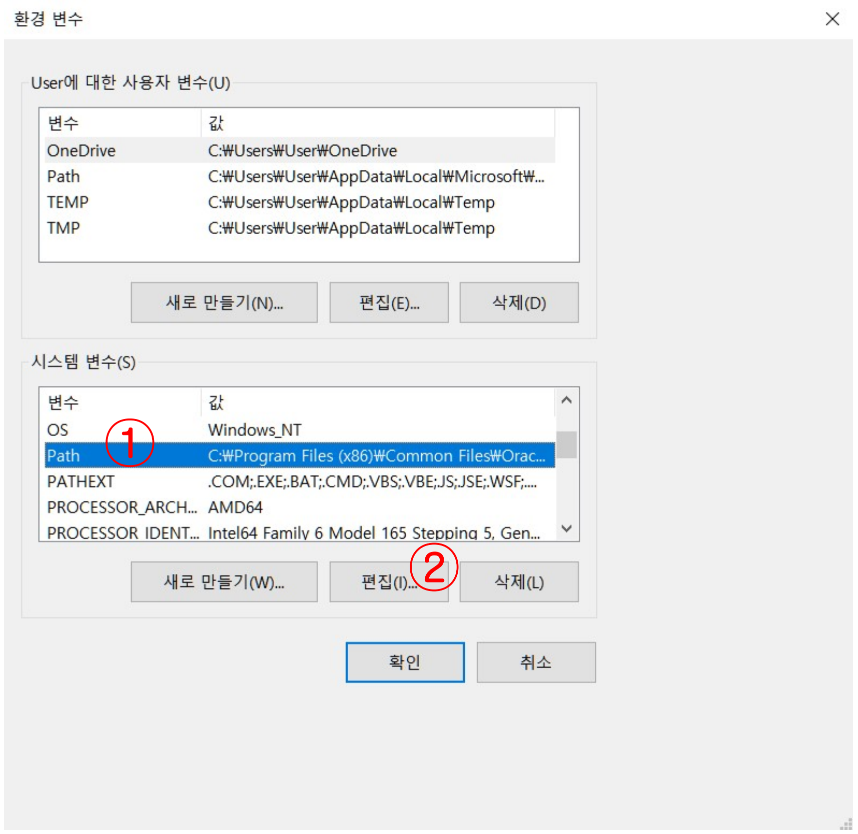


①: search and excute
'시스템 환경 변수 편집'


시스템 속성
컴퓨터 이름
하드웨어
고급
시스템 보호
원격
이 내용을 변경하려면 관리자로 로그온해야 합니다.
성능
시각 효과, 프로세서 일정, 메모리 사용 및 가상 메모리
설정(S)...
사용자 프로필
사용자 로그인에 관련된 바탕 화면 설정
설정(E)...
시작 및 복구
시스템 시작, 시스템 오류 및 디버깅 정보
설정(T)...
②
환경 변수(N)...
확인
취소
적용(A)

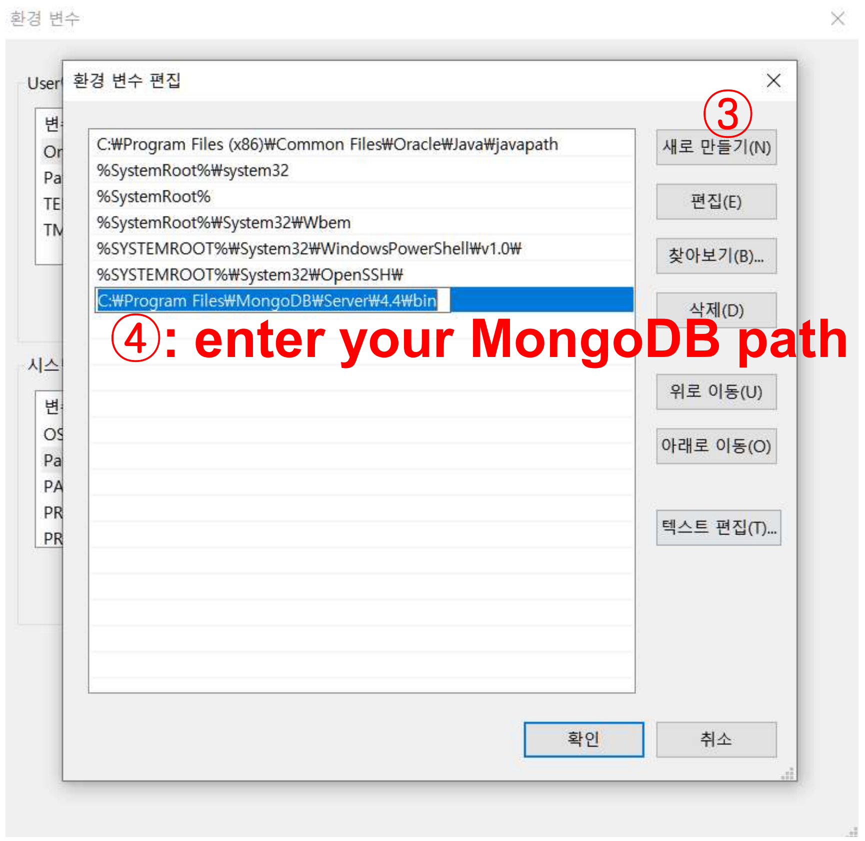
Practices

- Setup MongoDB
  - Set the environment variables



변수	값
OneDrive	C:₩Users₩User₩OneDrive
Path	C:₩Users₩User₩AppData₩Local₩Microsoft₩...
TEMP	C:₩Users₩User₩AppData₩Local₩Temp
TMP	C:₩Users₩User₩AppData₩Local₩Temp

변수	값
OS	Windows_NT
Path	C:₩Program Files (x86)₩Common Files₩Orac...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;...
PROCESSOR_ARCH...	AMD64
PROCESSOR_IDENT...	Intel64 Family 6 Model 165 Stepping 5, Gen...

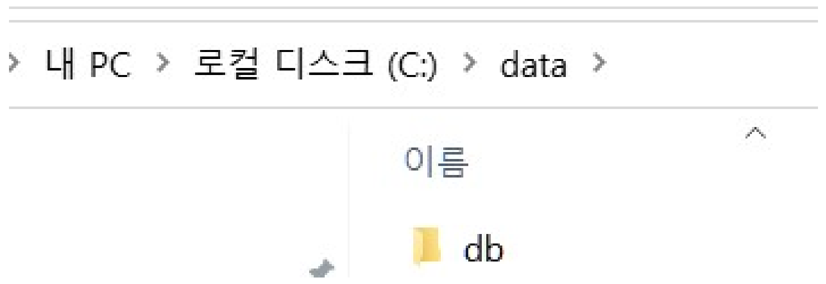


```
C:₩Program Files (x86)₩Common Files₩Oracle₩Java₩javapath
%SystemRoot%₩system32
%SystemRoot%
%SystemRoot%₩System32₩Wbem
%SYSTEMROOT%₩System32₩WindowsPowerShell₩v1.0₩
%SYSTEMROOT%₩System32₩OpenSSH₩
C:₩Program Files₩MongoDB₩Server₩4.4₩bin
```

④: enter your MongoDB path

Practices

- Setup MongoDB
  - Make a directory → C:₩data₩db
    - It saves all of the *CRUD activities in the MongoDB server

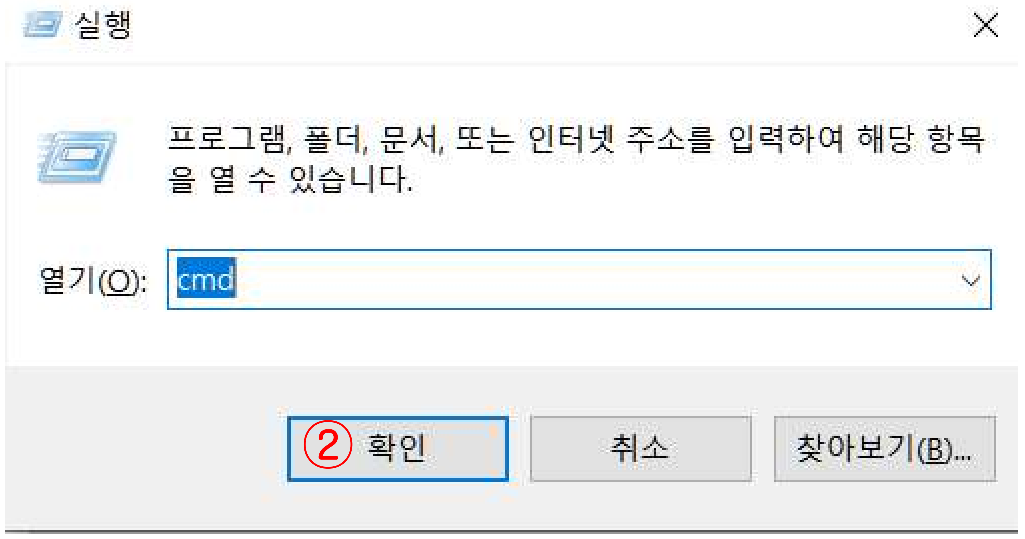


*CRUD: create, read, update, delete

Practices

- Setup MongoDB
  - Check out the installation

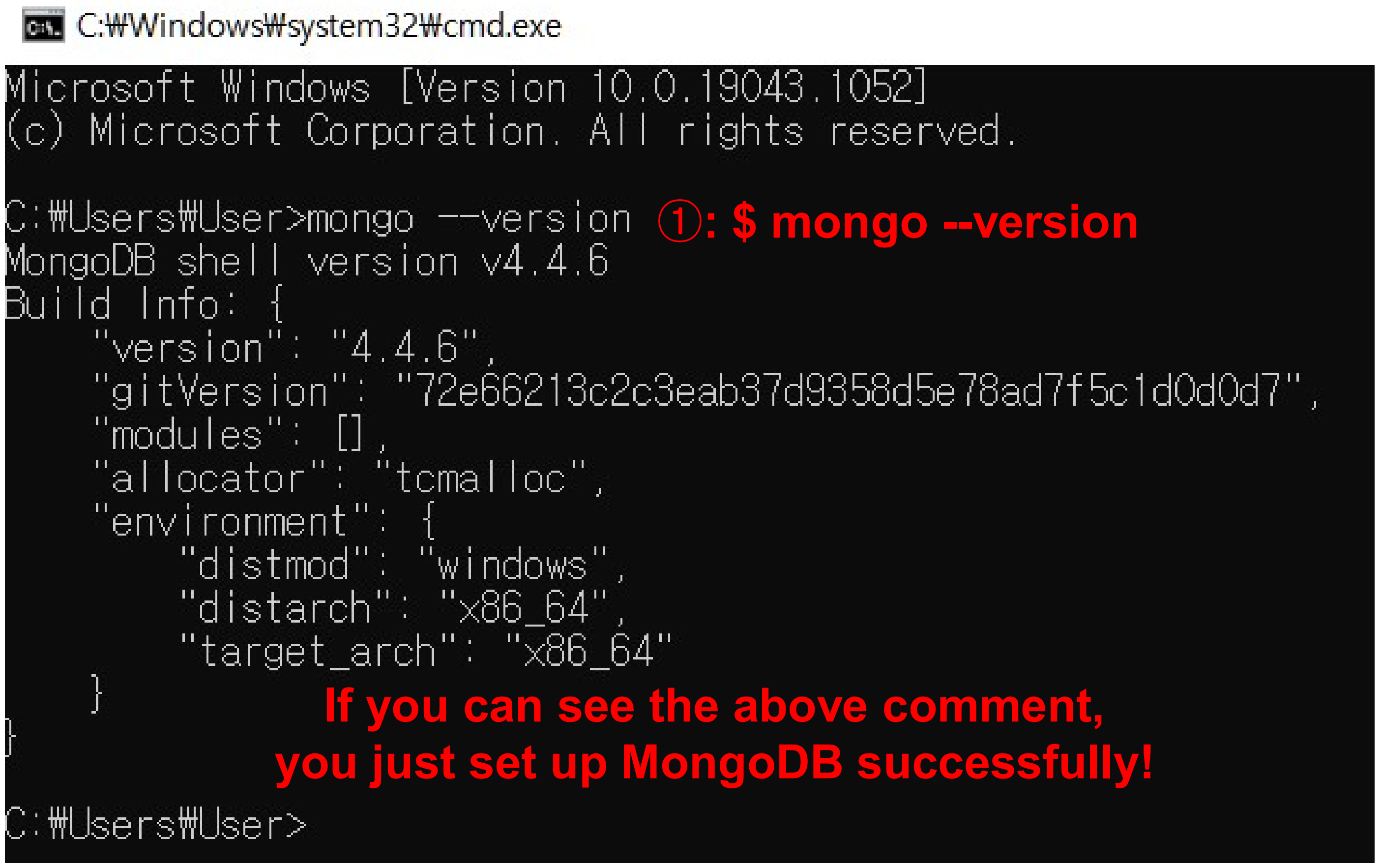
①: win+r



프로그램, 폴더, 문서, 또는 인터넷 주소를 입력하여 해당 항목을 열 수 있습니다.

Practices

- Setup MongoDB
  - Check out the installation



C:₩Windows₩system32₩cmd.exe

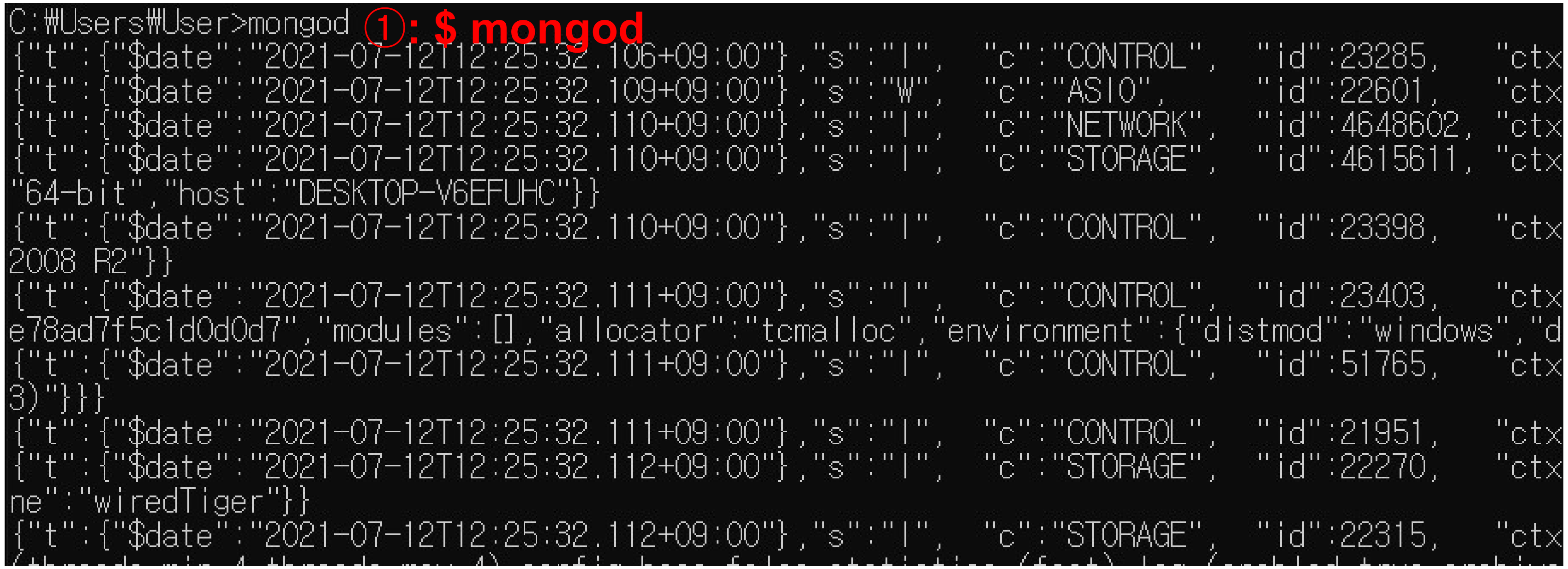
```
Microsoft Windows [Version 10.0.19043.1052]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩User>mongo --version
MongoDB shell version v4.4.6
Build Info: {
    "version": "4.4.6",
    "gitVersion": "72e66213c2c3eab37d9358d5e78ad7f5c1d0d0d7",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}

C:₩Users₩User>
```

①: $ mongo --version

If you can see the above comment, you just set up MongoDB successfully!

Practices

- Setup MongoDB
  - Activate a server

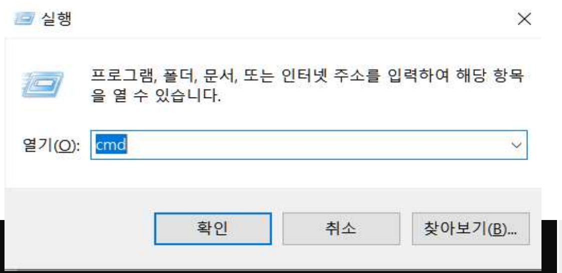


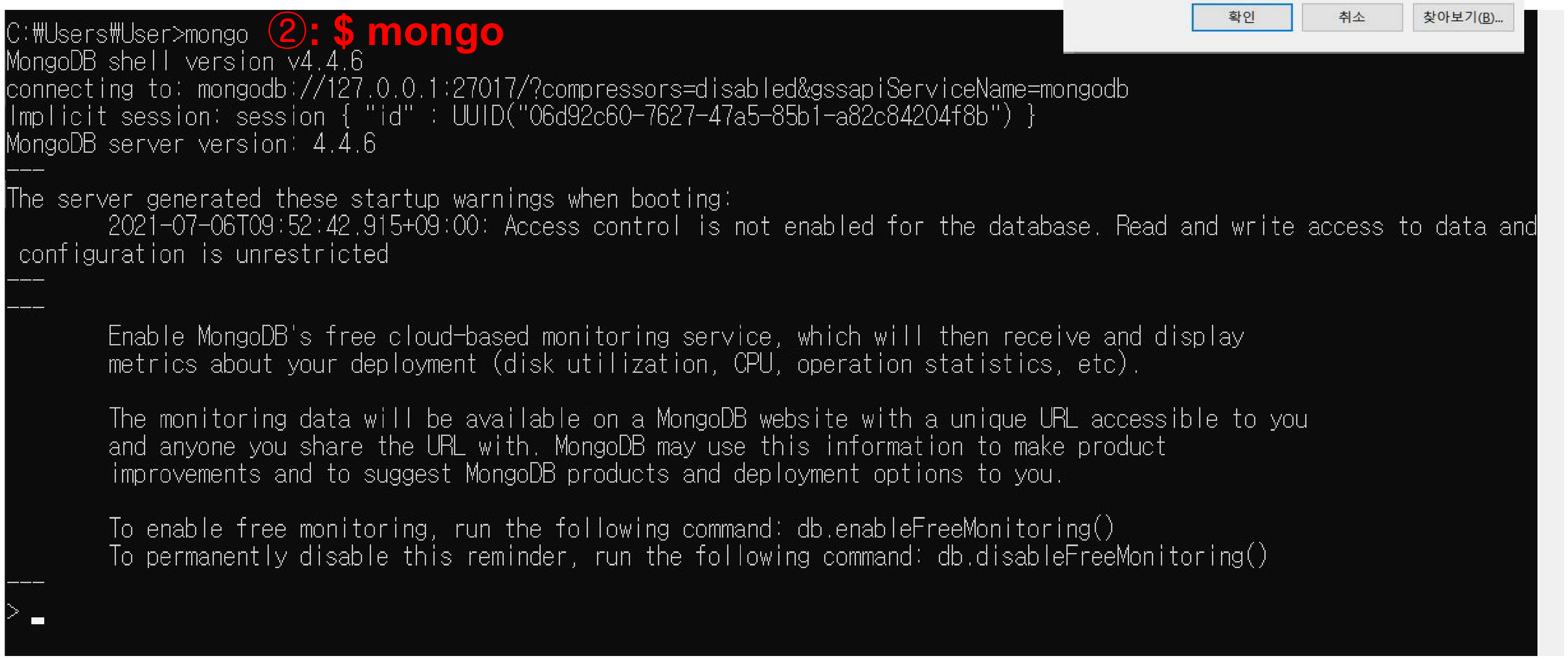
```
C:₩Users₩User>mongod ①: $ mongod
{"t":{"$date":"2021-07-12T12:25:32.106+09:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.109+09:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx
"64-bit","host":"DESKTOP-V6EFUHC"}}
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx
2008 R2"}}
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx
e78ad7f5c1d0d0d7","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","d
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx
3)"}}}
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.112+09:00"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx
ne":"wiredTiger"}}
{"t":{"$date":"2021-07-12T12:25:32.112+09:00"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx
```

Practices

- Look around MongoDB shell
  - Enter the server

①: open another command window


실행
프로그램, 폴더, 문서, 또는 인터넷 주소를 입력하여 해당 항목을 열 수 있습니다.
열기(O):
cmd
확인
취소
찾아보기(B)...



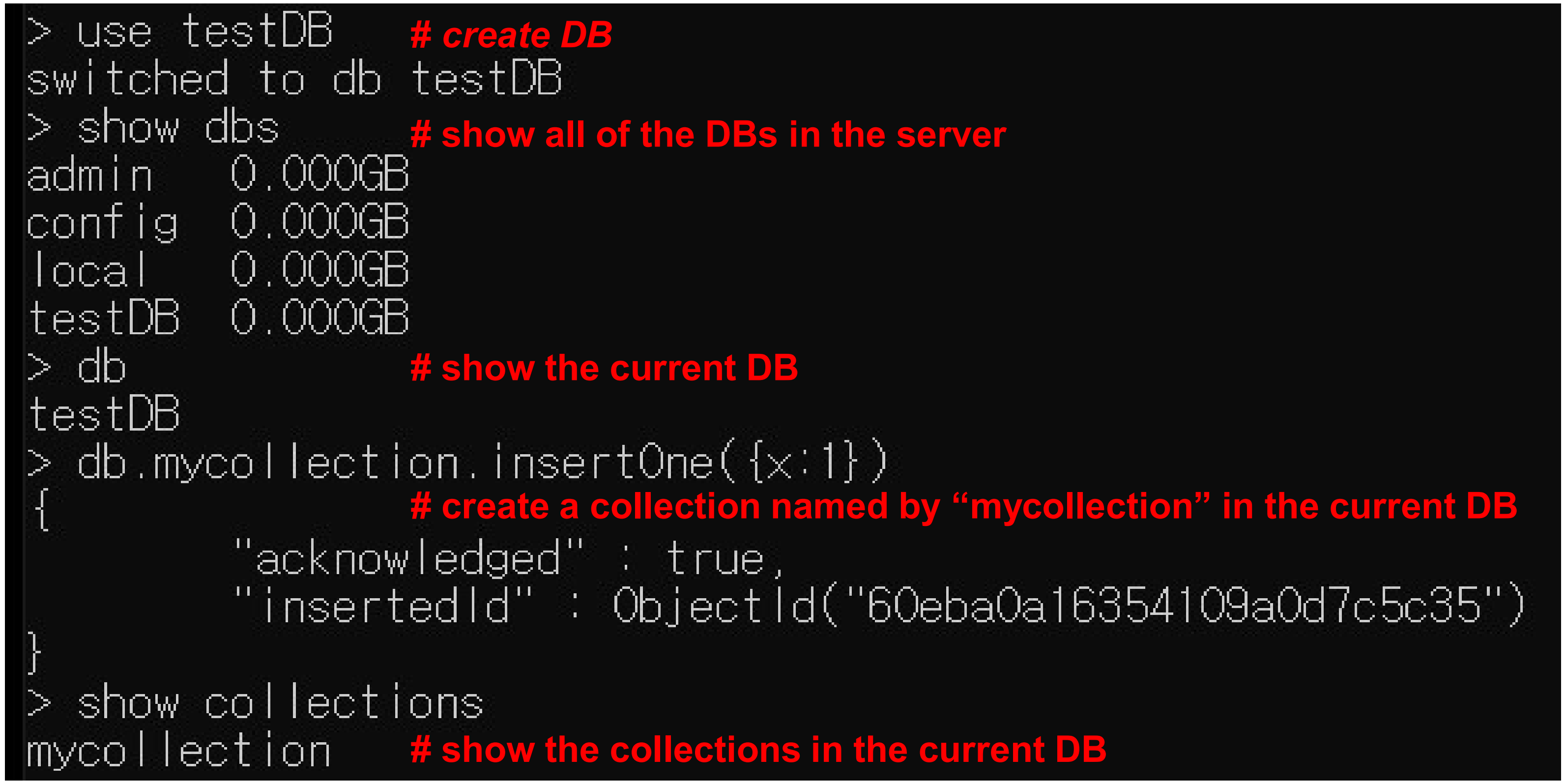
②: $ mongo

```
C:₩Users₩User>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("06d92c60-7627-47a5-85b1-a82c84204f8b") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
        2021-07-06T09:52:42.915+09:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> _
```

Practices

"acknowledged" : true → 데이터 생성 성공
"insertedID" : ObjectId → Primary Key

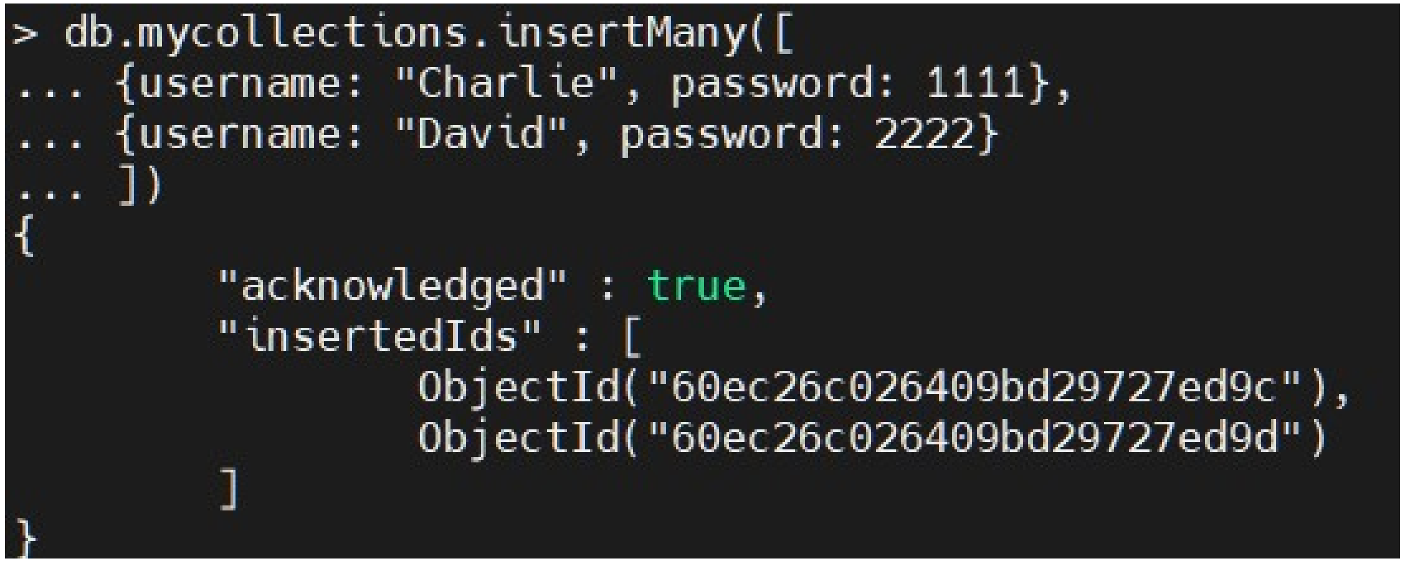
- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> use testDB                # create DB
switched to db testDB
> show dbs                  # show all of the DBs in the server
admin   0.000GB
config  0.000GB
local   0.000GB
testDB  0.000GB
> db                        # show the current DB
testDB
> db.mycollection.insertOne({x:1})
{                           # create a collection named by "mycollection" in the current DB
        "acknowledged" : true,
        "insertedId" : ObjectId("60eba0a16354109a0d7c5c35")
}
> show collections
mycollection                # show the collections in the current DB
```

Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete

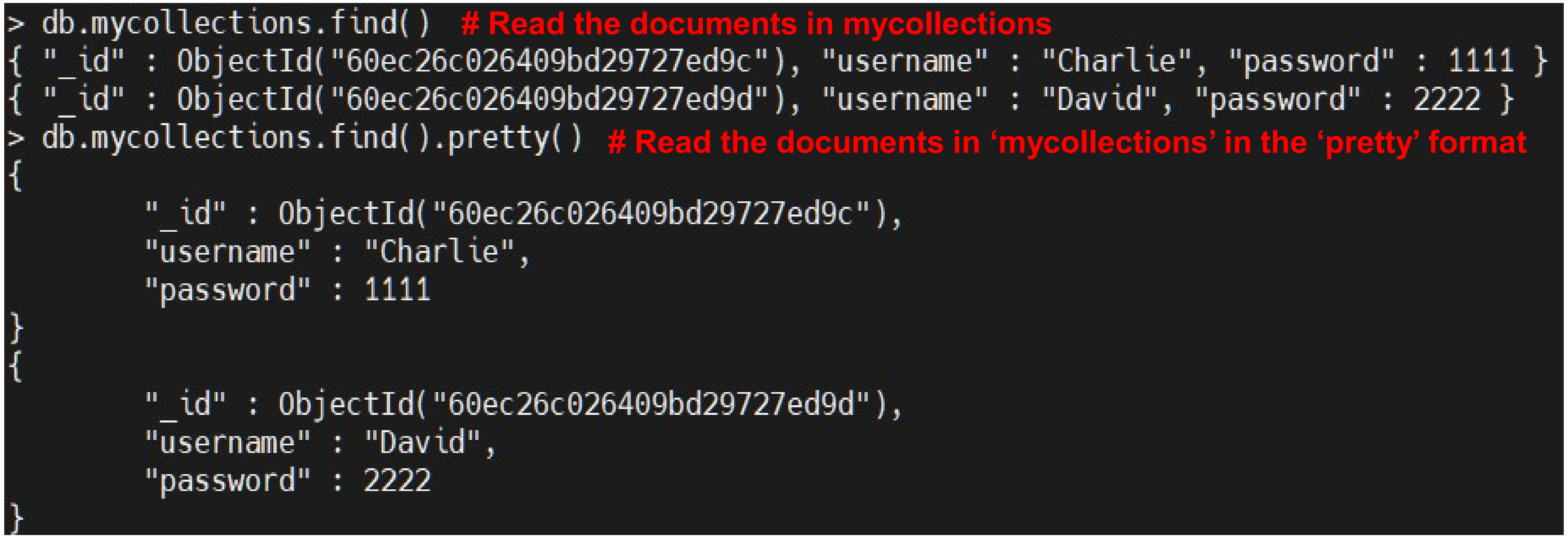


```
> db.mycollections.insertMany([
... {username: "Charlie", password: 1111},
... {username: "David", password: 2222}
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60ec26c026409bd29727ed9c"),
                ObjectId("60ec26c026409bd29727ed9d")
        ]
}
```

# Create a collection named by ‘mycollections’ and add documents in the collection

Practices

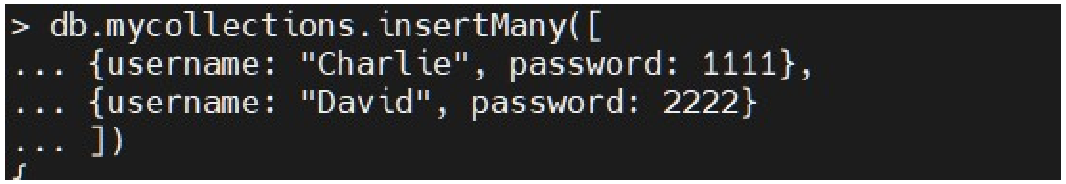
- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> db.mycollections.find()  # Read the documents in mycollections
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec26c026409bd29727ed9d"), "username" : "David", "password" : 2222 }
> db.mycollections.find().pretty()  # Read the documents in ‘mycollections’ in the ‘pretty’ format
{
        "_id" : ObjectId("60ec26c026409bd29727ed9c"),
        "username" : "Charlie",
        "password" : 1111
}
{
        "_id" : ObjectId("60ec26c026409bd29727ed9d"),
        "username" : "David",
        "password" : 2222
}
```

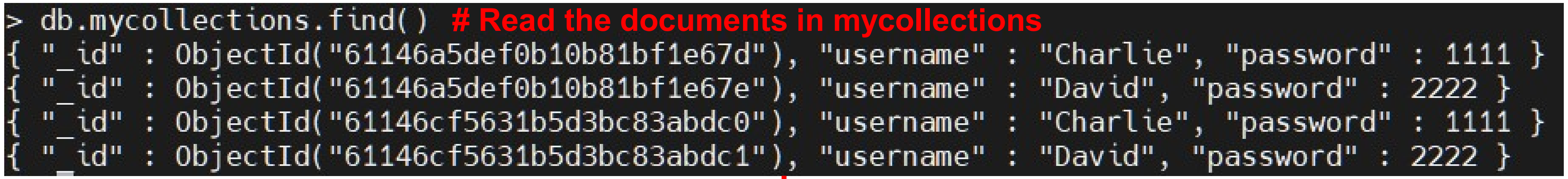
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



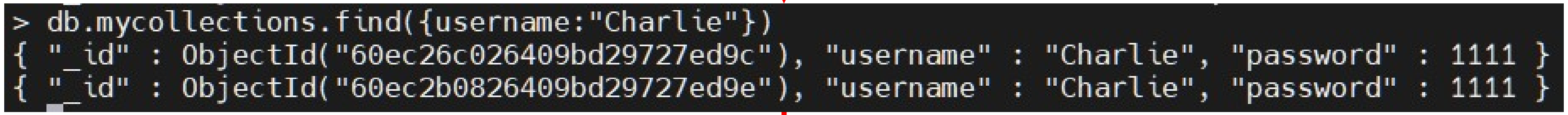
```
> db.mycollections.insertMany([
... {username: "Charlie", password: 1111},
... {username: "David", password: 2222}
... ])
```

# Create additional documents in mycollections



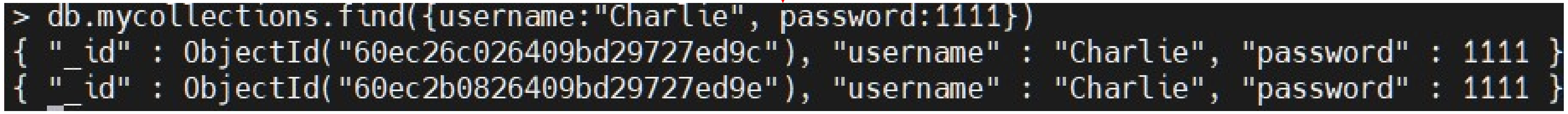
```
> db.mycollections.find() # Read the documents in mycollections
{ "_id" : ObjectId("61146a5def0b10b81bf1e67d"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146a5def0b10b81bf1e67e"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc0"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc1"), "username" : "David", "password" : 2222 }
```

# Read the documents whose username is "Charlie"



```
> db.mycollections.find({username:"Charlie"})
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9e"), "username" : "Charlie", "password" : 1111 }
```

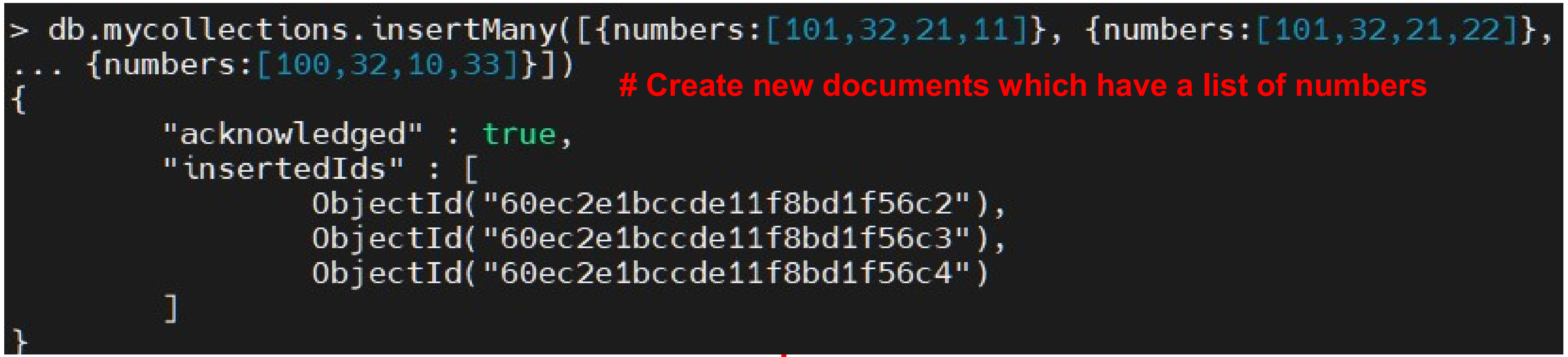
# Read the documents whose username is "Charlie" and password is 1111



```
> db.mycollections.find({username:"Charlie", password:1111})
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9e"), "username" : "Charlie", "password" : 1111 }
```

Practices

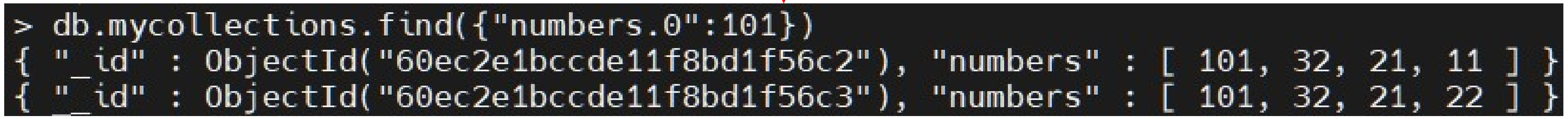
- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> db.mycollections.insertMany([{numbers:[101,32,21,11]}, {numbers:[101,32,21,22]},
... {numbers:[100,32,10,33]}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60ec2e1bccde11f8bd1f56c2"),
                ObjectId("60ec2e1bccde11f8bd1f56c3"),
                ObjectId("60ec2e1bccde11f8bd1f56c4")
        ]
}
```

# Create new documents which have a list of numbers

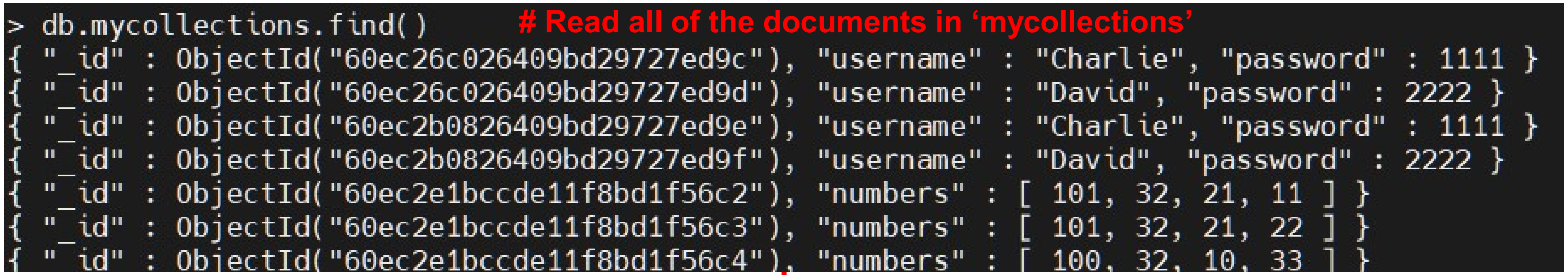
# Read the documents whose first number is 101



```
> db.mycollections.find({"numbers.0":101})
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
```

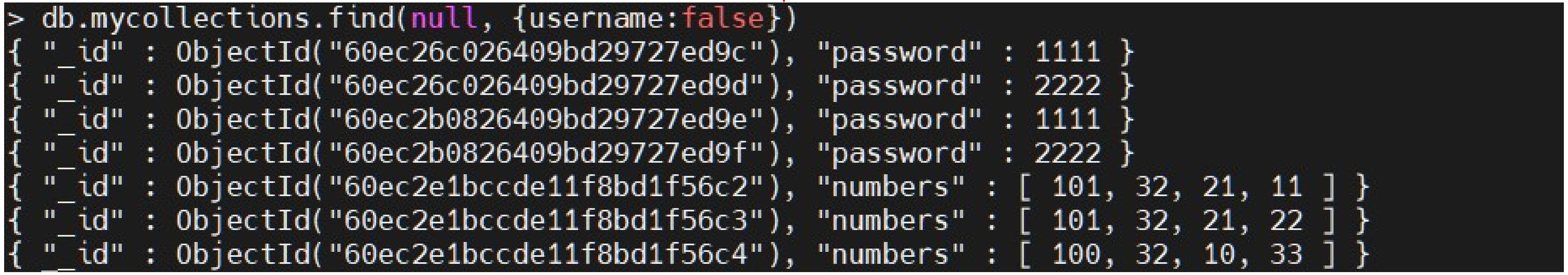
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> db.mycollections.find()        # Read all of the documents in 'mycollections'
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec26c026409bd29727ed9d"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9e"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9f"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

# Read the documents in the DB except for the field 'username'



```
> db.mycollections.find(null, {username:false})
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "password" : 1111 }
{ "_id" : ObjectId("60ec26c026409bd29727ed9d"), "password" : 2222 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9e"), "password" : 1111 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9f"), "password" : 2222 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

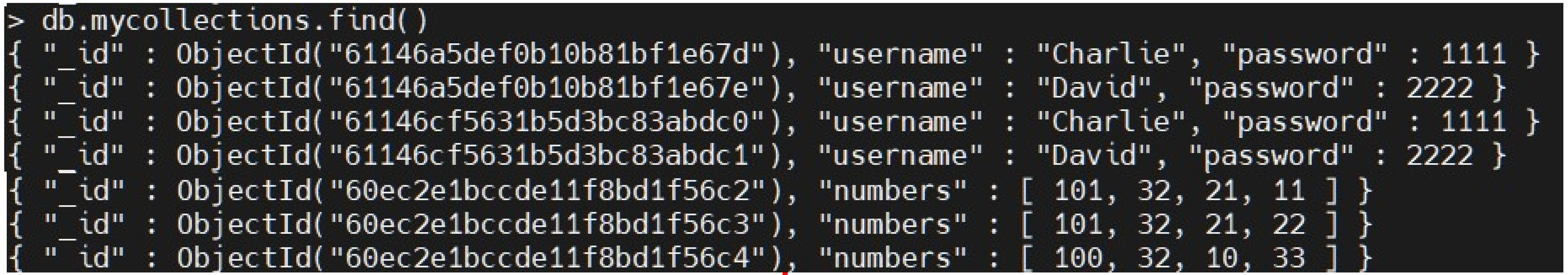
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete

Operators	**Description**
$currentDate	Sets the value of a field to current date, either as a Date or a Timestamp.
$inc	Increments the value of the field by the specified amount.
$min	Only updates the field if the specified value is less than the existing field value.
$max	Only updates the field if the specified value is greater than the existing field value.
$mul	Multiplies the value of the field by the specified amount.
$rename	Renames a field.
$set	Sets the value of a field in a document.
$setOnInsert	Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
$unset	Removes the specified field from a document.

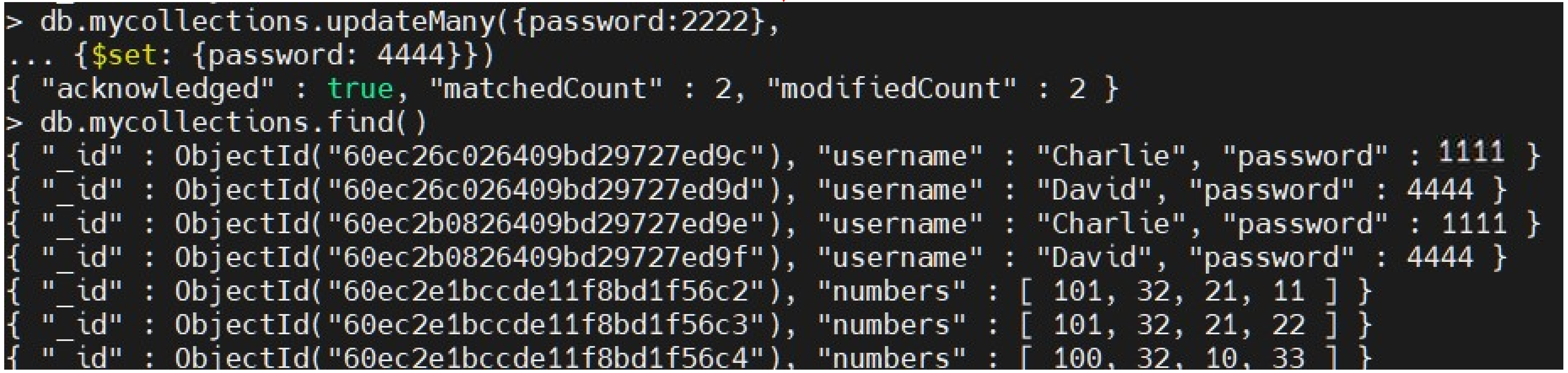
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> db.mycollections.find()
{ "_id" : ObjectId("61146a5def0b10b81bf1e67d"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146a5def0b10b81bf1e67e"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc0"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc1"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

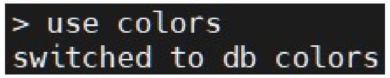
# Change the password: 2222 → 4444



```
> db.mycollections.updateMany({password:2222},
... {$set: {password: 4444}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.mycollections.find()
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec26c026409bd29727ed9d"), "username" : "David", "password" : 4444 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9e"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("60ec2b0826409bd29727ed9f"), "username" : "David", "password" : 4444 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

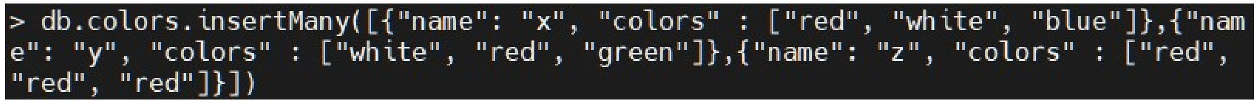
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



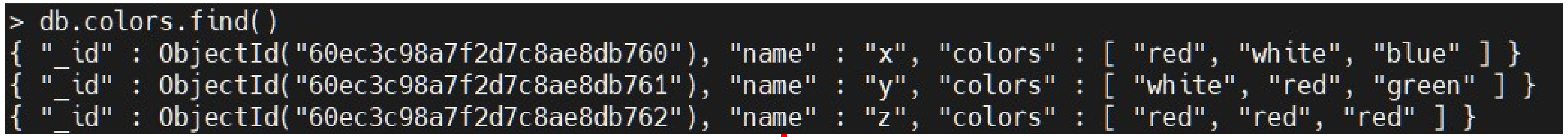
```
> use colors
switched to db colors
```

# Create a new DB named by "colors"



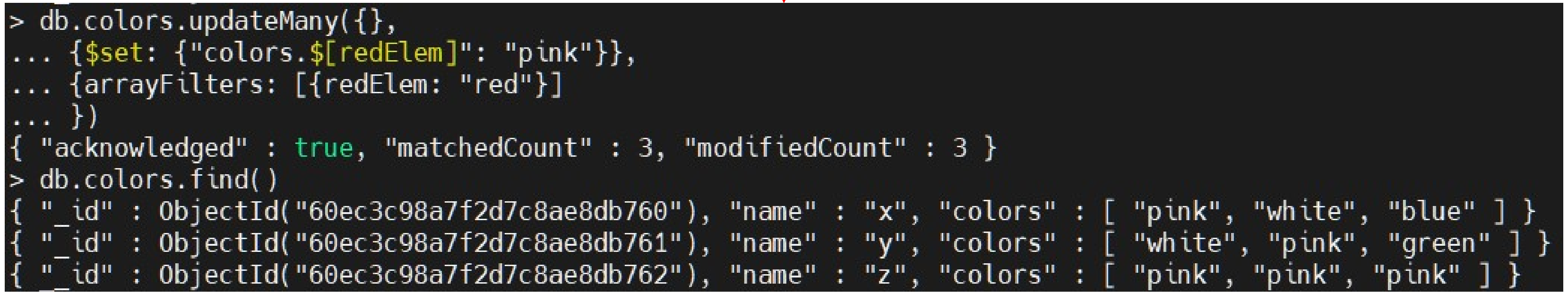
```
> db.colors.insertMany([{"name": "x", "colors" : ["red", "white", "blue"]},{"nam
e": "y", "colors" : ["white", "red", "green"]},{"name": "z", "colors" : ["red",
"red", "red"]}])
```

# Create documents in "colors"



```
> db.colors.find()
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db760"), "name" : "x", "colors" : [ "red", "white", "blue" ] }
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db761"), "name" : "y", "colors" : [ "white", "red", "green" ] }
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db762"), "name" : "z", "colors" : [ "red", "red", "red" ] }
```

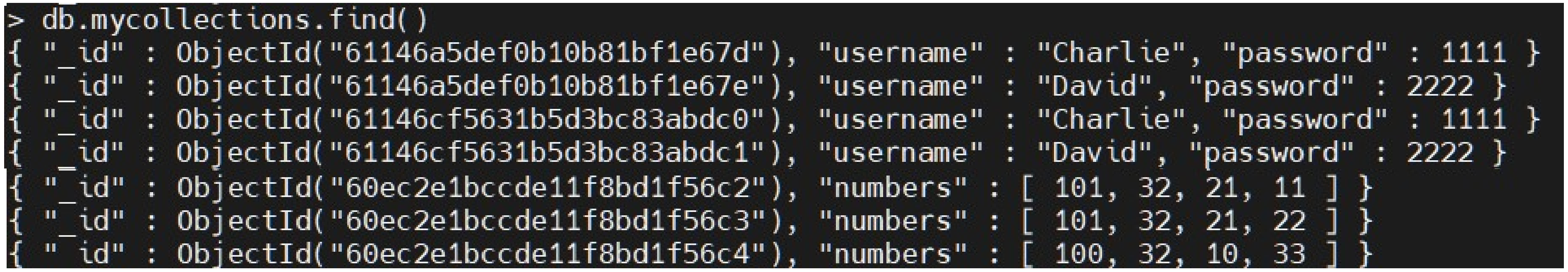
# Change the color: "red" → "pink"



```
> db.colors.updateMany({},
... {$set: {"colors.$[redElem]": "pink"}},
... {arrayFilters: [{redElem: "red"}]
... })
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.colors.find()
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db760"), "name" : "x", "colors" : [ "pink", "white", "blue" ] }
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db761"), "name" : "y", "colors" : [ "white", "pink", "green" ] }
{ "_id" : ObjectId("60ec3c98a7f2d7c8ae8db762"), "name" : "z", "colors" : [ "pink", "pink", "pink" ] }
```

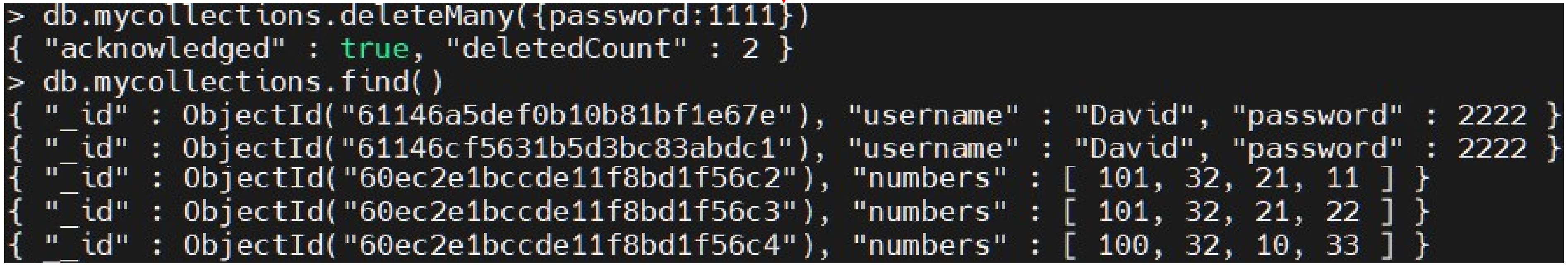
Practices

- Look around MongoDB shell
  - CRUD: Create, Read, Update, Delete



```
> db.mycollections.find()
{ "_id" : ObjectId("61146a5def0b10b81bf1e67d"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146a5def0b10b81bf1e67e"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc0"), "username" : "Charlie", "password" : 1111 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc1"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

# Delete the documents having password 1111

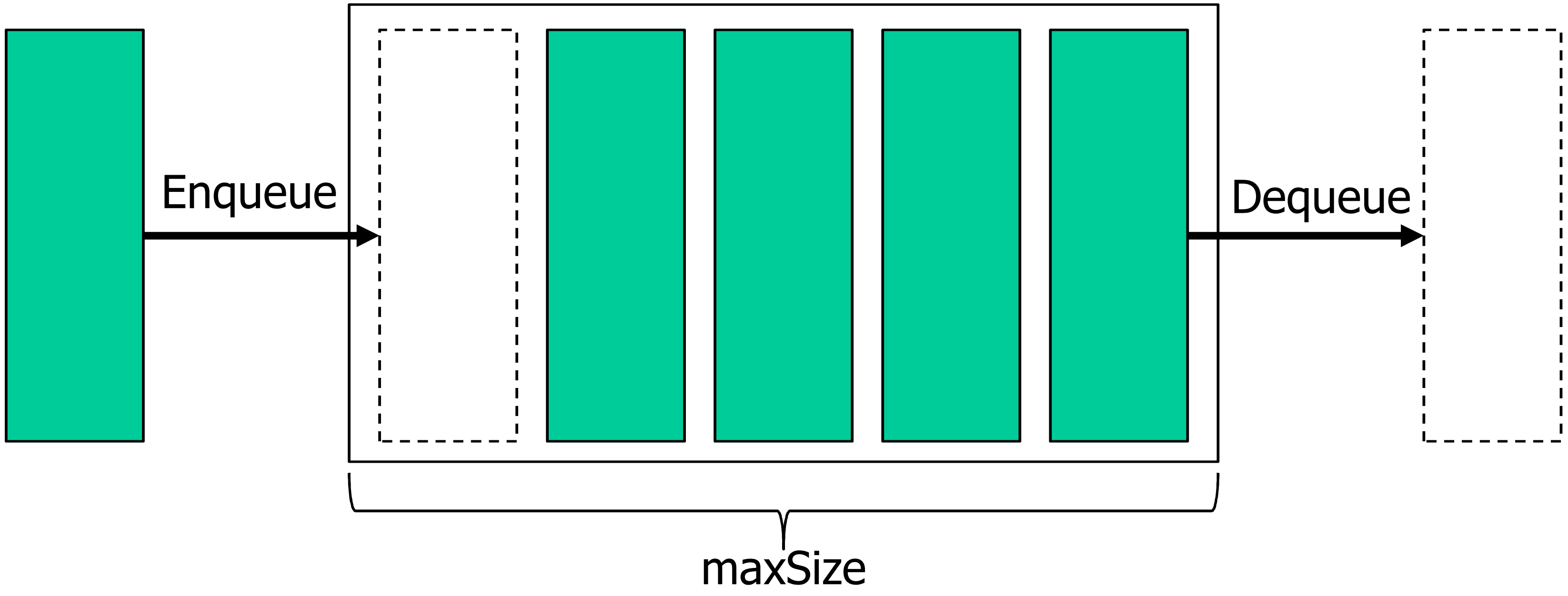


```
> db.mycollections.deleteMany({password:1111})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.mycollections.find()
{ "_id" : ObjectId("61146a5def0b10b81bf1e67e"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("61146cf5631b5d3bc83abdc1"), "username" : "David", "password" : 2222 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

Capped collections는 고정 사이즈 collections으로 빠른 처리량이 필요한 업무에서 사용되며, insert 와 retrieve 가 insertion 의 순서대로 이루어지는 특징을 가지고 있다. 이러한 형태는 circular buffers (queue)의 특징을 가지고 있으며, 가장 오래된 documents을 overwrite 하는 형태로 새로운 document가 만들어 지게 된다.

- Look around MongoDB shell
  - Capped collection

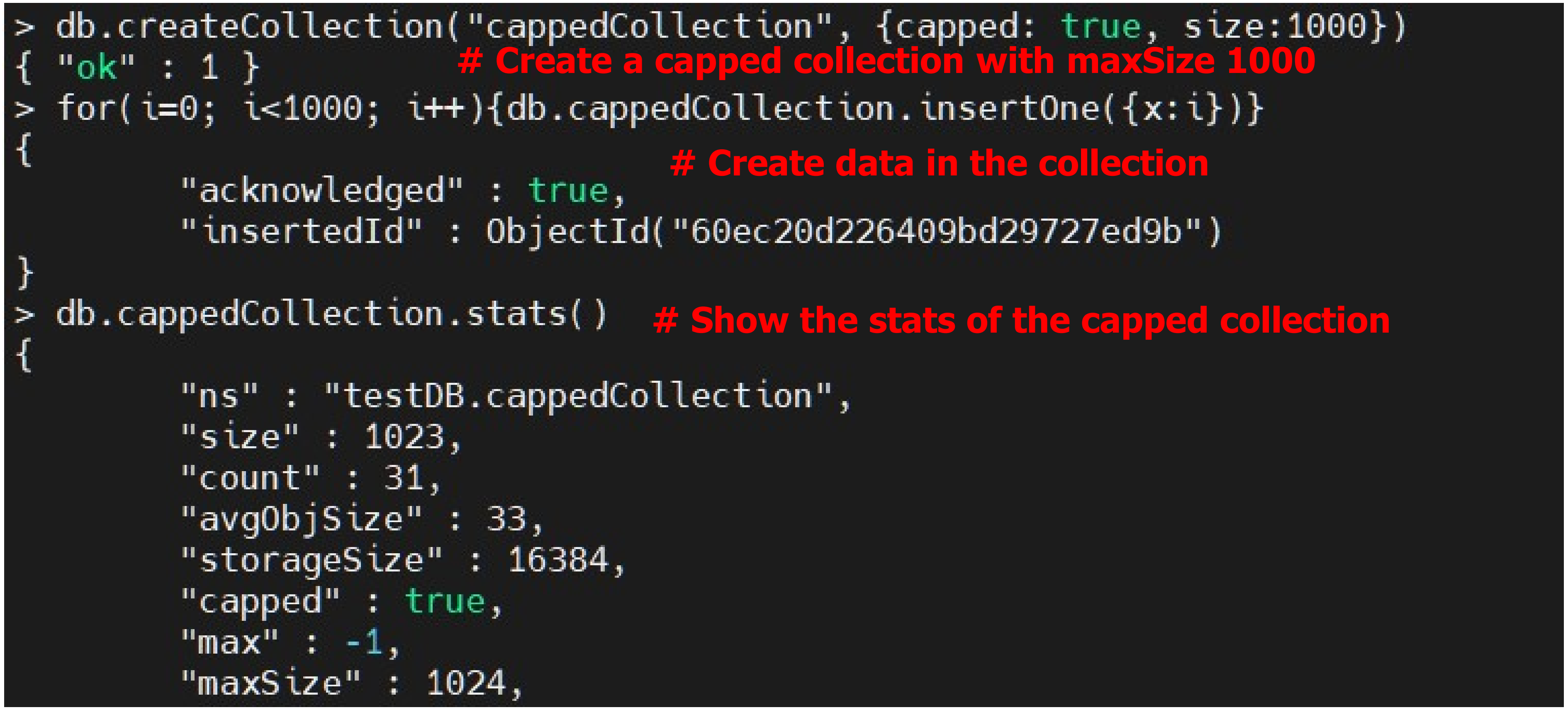
```
$db.createCollection(<collection name>, {capped: true, size: {maxSize})
```



Practices

- Look around MongoDB shell
  - Capped collection

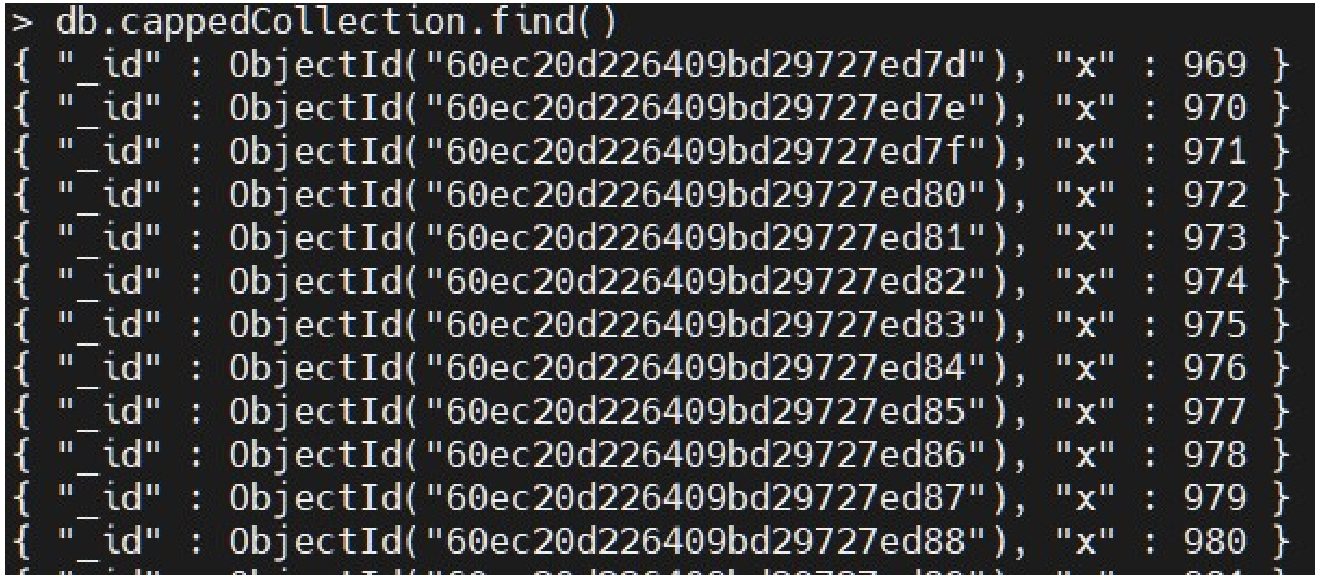
maxSize는 입력값과 가장 가까운 256배수로 설정



```
> db.createCollection("cappedCollection", {capped: true, size:1000})
{ "ok" : 1 }            # Create a capped collection with maxSize 1000
> for(i=0; i<1000; i++){db.cappedCollection.insertOne({x:i})}
{
                                     # Create data in the collection
        "acknowledged" : true,
        "insertedId" : ObjectId("60ec20d226409bd29727ed9b")
}
> db.cappedCollection.stats()   # Show the stats of the capped collection
{
        "ns" : "testDB.cappedCollection",
        "size" : 1023,
        "count" : 31,
        "avgObjSize" : 33,
        "storageSize" : 16384,
        "capped" : true,
        "max" : -1,
        "maxSize" : 1024,
```

Practices

- Look around MongoDB shell
  - Capped collection



```
> db.cappedCollection.find()
{ "_id" : ObjectId("60ec20d226409bd29727ed7d"), "x" : 969 }
{ "_id" : ObjectId("60ec20d226409bd29727ed7e"), "x" : 970 }
{ "_id" : ObjectId("60ec20d226409bd29727ed7f"), "x" : 971 }
{ "_id" : ObjectId("60ec20d226409bd29727ed80"), "x" : 972 }
{ "_id" : ObjectId("60ec20d226409bd29727ed81"), "x" : 973 }
{ "_id" : ObjectId("60ec20d226409bd29727ed82"), "x" : 974 }
{ "_id" : ObjectId("60ec20d226409bd29727ed83"), "x" : 975 }
{ "_id" : ObjectId("60ec20d226409bd29727ed84"), "x" : 976 }
{ "_id" : ObjectId("60ec20d226409bd29727ed85"), "x" : 977 }
{ "_id" : ObjectId("60ec20d226409bd29727ed86"), "x" : 978 }
{ "_id" : ObjectId("60ec20d226409bd29727ed87"), "x" : 979 }
{ "_id" : ObjectId("60ec20d226409bd29727ed88"), "x" : 980 }
```

Dequeue: 1~968

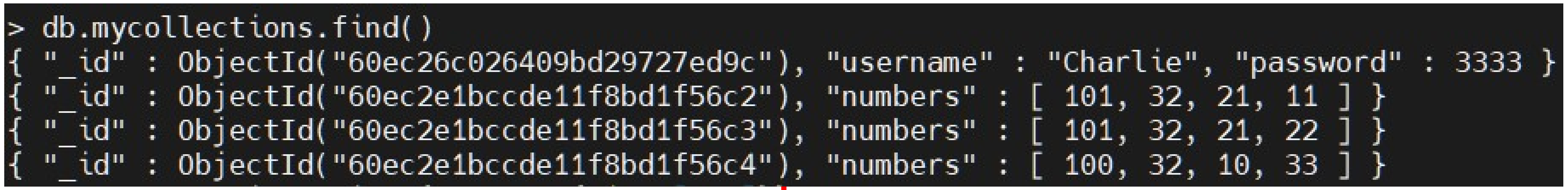
Practices

- Query operator – Comparison

Name	**Description**
$eq	(equal) Matches values that **are equal to** a specified value.
$ne	(not equal) Matches all values that **are not equal to** a specified value.
$gt	(greater than) Matches values that **are greater than** a specified value.
$gte	(greater than or equal) Matches values that **are greater than or equal** to a specified value.
$lt	(less than) Matches values that **are less than** a specified value.
$lte	(less than or equal) Matches values that **are less than or equal** to a specified value.
$in	(in) Matches **any of the values specified in an array**.
$nin	(not in) Matches **none of the values specified in an array**.

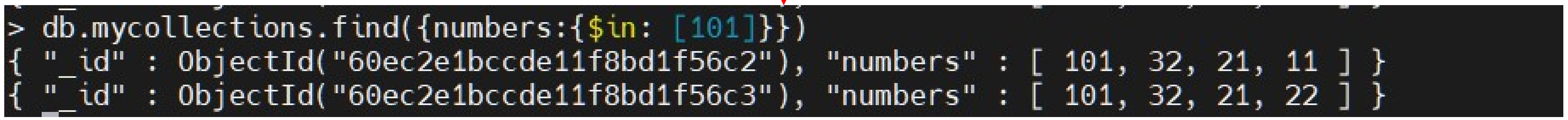
Practices

- Query operator – Comparison



```
> db.mycollections.find()
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 3333 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

# Find a document that contains numbers 101



```
> db.mycollections.find({numbers:{$in: [101]}})
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
```

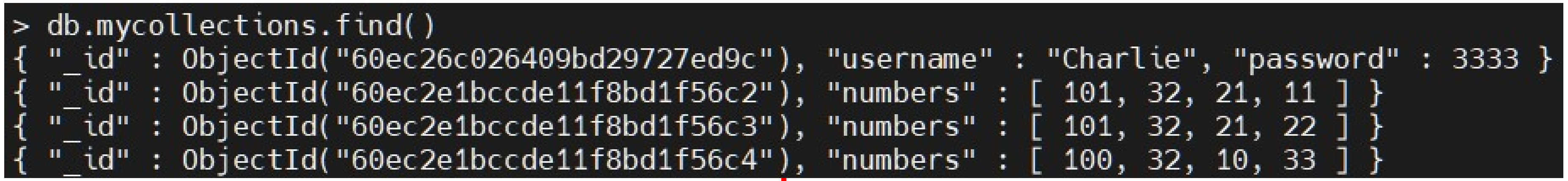
Practices

- Query operator – logical

Name	**Description**
$and	Joins query clauses with a logical AND **returns all documents that match the conditions of both clauses**.
$not	Inverts the effect of a query expression and **returns documents that do not match the query expression**.
$nor	Joins query clauses with a logical NOR **returns all documents that fail to match both clauses**.
$or	Joins query clauses with a logical OR **returns all documents that match the conditions of either clause**.

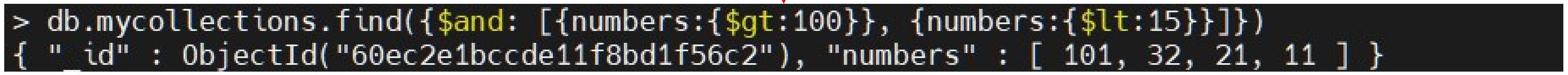
Practices

- Query operator – logical



```
> db.mycollections.find()
{ "_id" : ObjectId("60ec26c026409bd29727ed9c"), "username" : "Charlie", "password" : 3333 }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c3"), "numbers" : [ 101, 32, 21, 22 ] }
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c4"), "numbers" : [ 100, 32, 10, 33 ] }
```

# Find a document that contains numbers are greater than 100 and less than 15

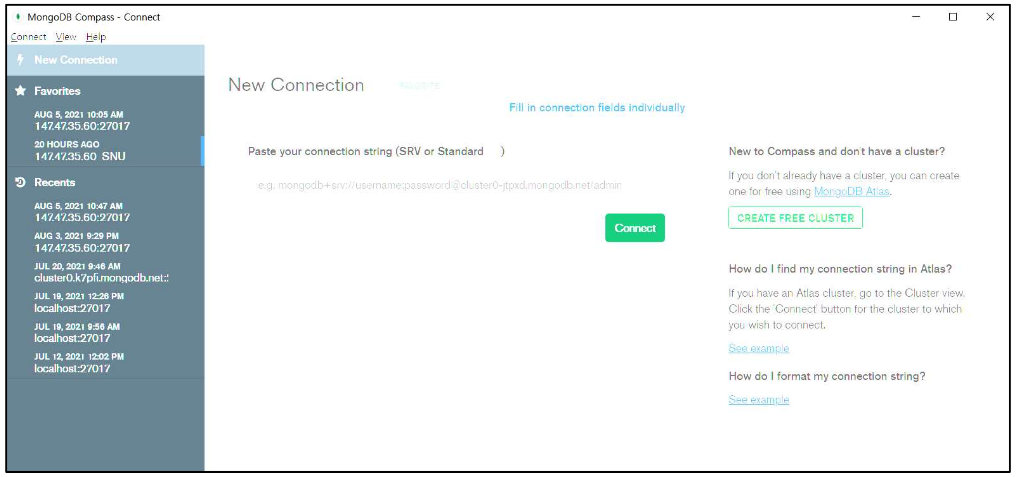


```
> db.mycollections.find({$and: [{numbers:{$gt:100}}, {numbers:{$lt:15}}]})
{ "_id" : ObjectId("60ec2e1bccde11f8bd1f56c2"), "numbers" : [ 101, 32, 21, 11 ] }
```

Practices

- Query operator – others
  https://docs.mongodb.com/manual/reference/operator/

- MongoDB Compass (MongoDB용 GUI)
  https://docs.mongodb.com/compass/current/



AUG 5, 2021 10:05 AM
147.47.35.60:27017
20 HOURS AGO
147.47.35.60 SNU

AUG 5, 2021 10:47 AM
147.47.35.60:27017
AUG 3, 2021 9:29 PM
147.47.35.60:27017
JUL 20, 2021 9:46 AM
cluster0.k7pfi.mongodb.net:
JUL 19, 2021 12:26 PM
localhost:27017
JUL 19, 2021 9:56 AM
localhost:27017
JUL 12, 2021 12:02 PM
localhost:27017

New Connection

Fill in connection fields individually

Paste your connection string (SRV or Standard)

e.g. mongodb+srv://username:password@cluster0-jtpxd.mongodb.net/admin

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas.

How do I find my connection string in Atlas?

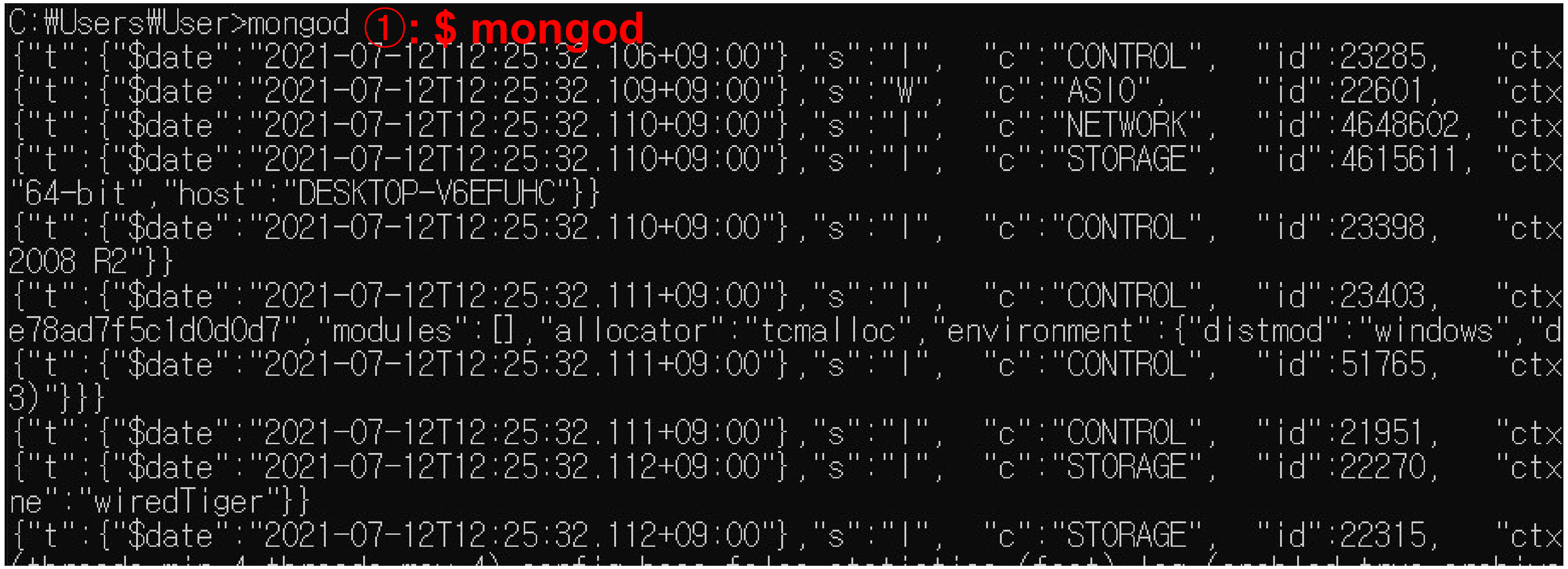
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.

How do I format my connection string?

NoSQL Exercises

Practices

- Setup MongoDB
  - Activate a server

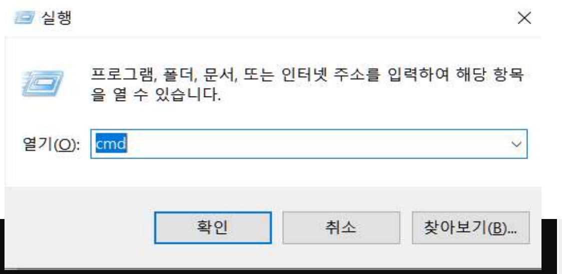


```
C:₩Users₩User>mongod
{"t":{"$date":"2021-07-12T12:25:32.106+09:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.109+09:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx
"64-bit","host":"DESKTOP-V6EFUHC"}}
{"t":{"$date":"2021-07-12T12:25:32.110+09:00"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx
2008 R2"}}
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx
e78ad7f5c1d0d0d7","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","d
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx
3)"}}}
{"t":{"$date":"2021-07-12T12:25:32.111+09:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx
{"t":{"$date":"2021-07-12T12:25:32.112+09:00"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx
ne":"wiredTiger"}}
{"t":{"$date":"2021-07-12T12:25:32.112+09:00"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx
```

Practices

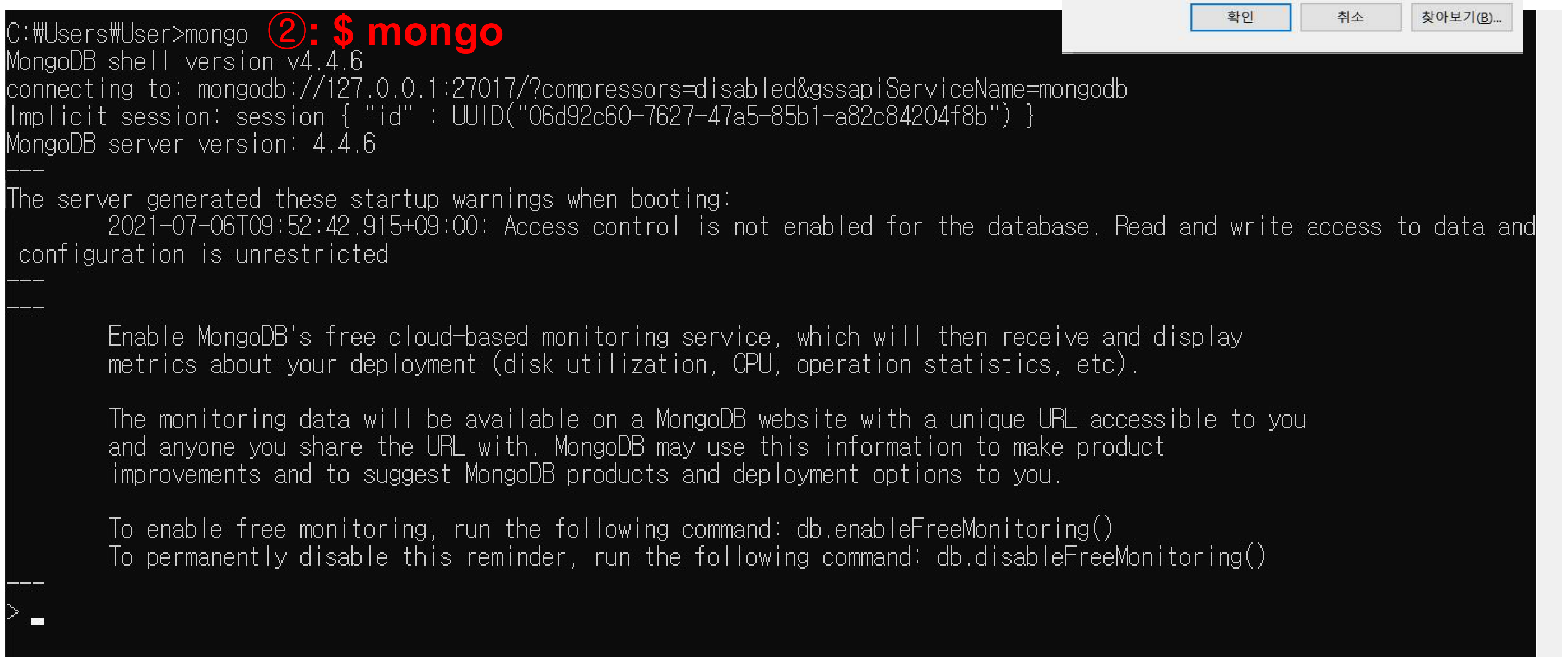
- Look around MongoDB shell
  - Enter the server

①: open another command window



프로그램, 폴더, 문서, 또는 인터넷 주소를 입력하여 해당 항목을 열 수 있습니다.

열기(O): cmd



②: $ mongo

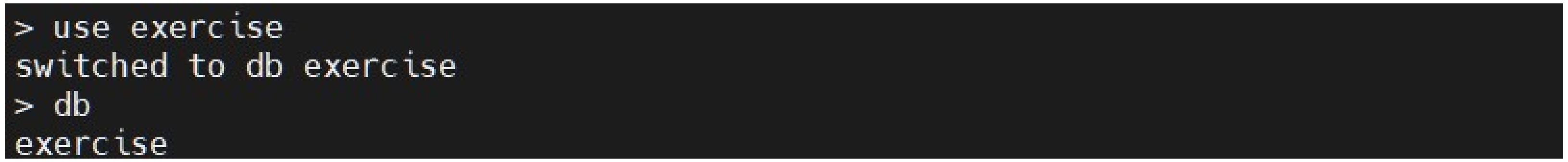
```
C:₩Users₩User>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("06d92c60-7627-47a5-85b1-a82c84204f8b") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
        2021-07-06T09:52:42.915+09:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> _
```

NoSQL Exercises

- Q1. Create DB named as ‘exercise’

NoSQL Exercises

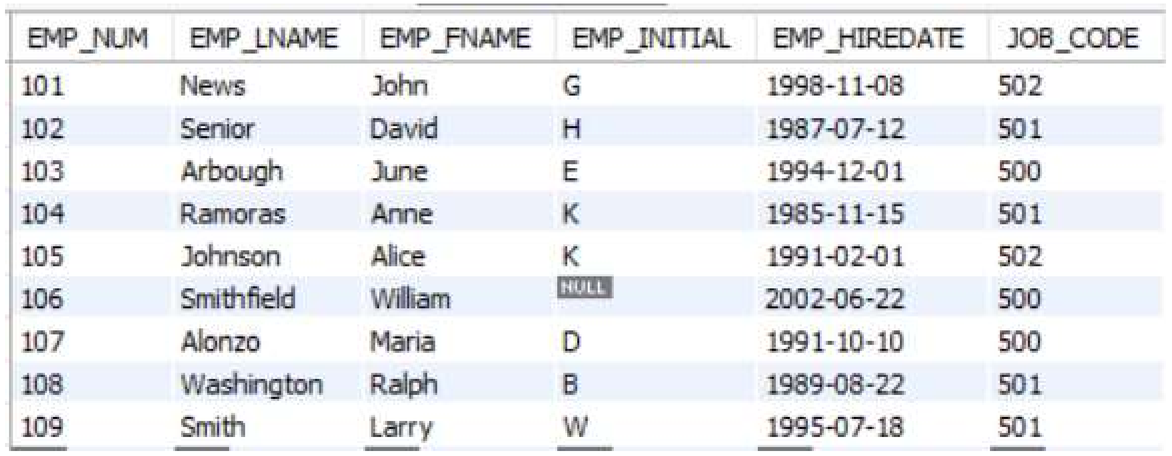
- A1.



```
> use exercise
switched to db exercise
> db
exercise
```

NoSQL Exercises

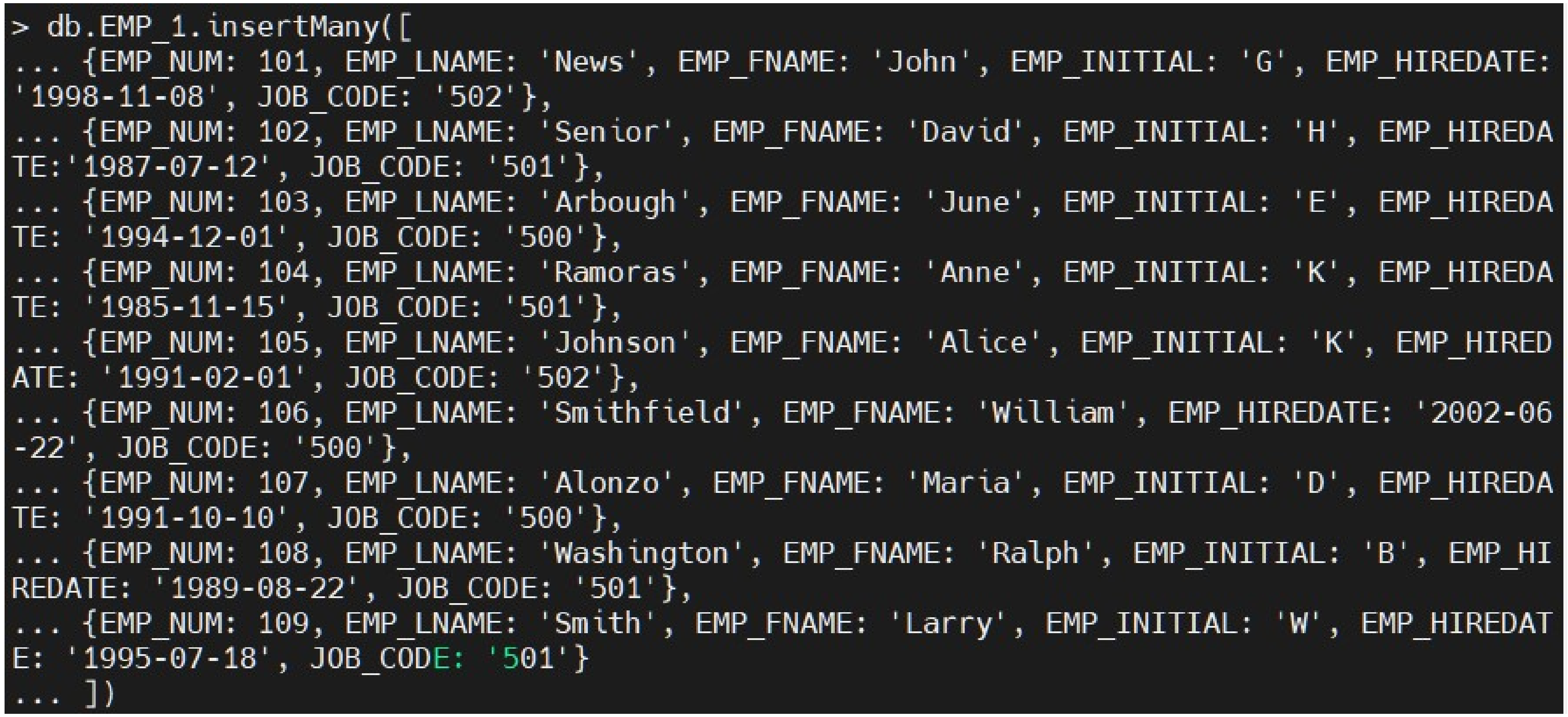
- Q2. Create a document named as 'EMP_1' and insert the collections including the below data



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	1998-11-08	502
102	Senior	David	H	1987-07-12	501
103	Arbough	June	E	1994-12-01	500
104	Ramoras	Anne	K	1985-11-15	501
105	Johnson	Alice	K	1991-02-01	502
106	Smithfield	William	NULL	2002-06-22	500
107	Alonzo	Maria	D	1991-10-10	500
108	Washington	Ralph	B	1989-08-22	501
109	Smith	Larry	W	1995-07-18	501

NoSQL Exercises

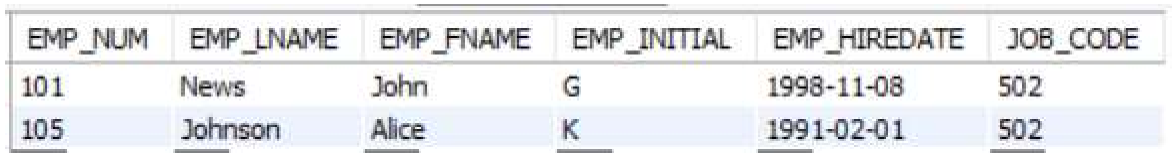
- A2.



```
> db.EMP_1.insertMany([
... {EMP_NUM: 101, EMP_LNAME: 'News', EMP_FNAME: 'John', EMP_INITIAL: 'G', EMP_HIREDATE:
'1998-11-08', JOB_CODE: '502'},
... {EMP_NUM: 102, EMP_LNAME: 'Senior', EMP_FNAME: 'David', EMP_INITIAL: 'H', EMP_HIREDA
TE:'1987-07-12', JOB_CODE: '501'},
... {EMP_NUM: 103, EMP_LNAME: 'Arbough', EMP_FNAME: 'June', EMP_INITIAL: 'E', EMP_HIREDA
TE: '1994-12-01', JOB_CODE: '500'},
... {EMP_NUM: 104, EMP_LNAME: 'Ramoras', EMP_FNAME: 'Anne', EMP_INITIAL: 'K', EMP_HIREDA
TE: '1985-11-15', JOB_CODE: '501'},
... {EMP_NUM: 105, EMP_LNAME: 'Johnson', EMP_FNAME: 'Alice', EMP_INITIAL: 'K', EMP_HIRED
ATE: '1991-02-01', JOB_CODE: '502'},
... {EMP_NUM: 106, EMP_LNAME: 'Smithfield', EMP_FNAME: 'William', EMP_HIREDATE: '2002-06
-22', JOB_CODE: '500'},
... {EMP_NUM: 107, EMP_LNAME: 'Alonzo', EMP_FNAME: 'Maria', EMP_INITIAL: 'D', EMP_HIREDA
TE: '1991-10-10', JOB_CODE: '500'},
... {EMP_NUM: 108, EMP_LNAME: 'Washington', EMP_FNAME: 'Ralph', EMP_INITIAL: 'B', EMP_HI
REDATE: '1989-08-22', JOB_CODE: '501'},
... {EMP_NUM: 109, EMP_LNAME: 'Smith', EMP_FNAME: 'Larry', EMP_INITIAL: 'W', EMP_HIREDAT
E: '1995-07-18', JOB_CODE: '501'}
... ])
```

NoSQL Exercises

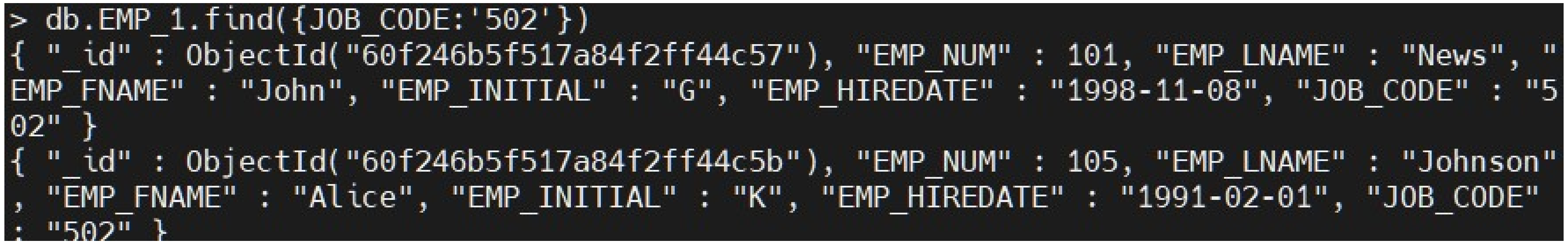
- Q3. Print all attributes for a job code of 502 from the table EMP_1



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	1998-11-08	502
105	Johnson	Alice	K	1991-02-01	502

NoSQL Exercises

- A3.



```
> db.EMP_1.find({JOB_CODE:'502'})
{ "_id" : ObjectId("60f246b5f517a84f2ff44c57"), "EMP_NUM" : 101, "EMP_LNAME" : "News", "
EMP_FNAME" : "John", "EMP_INITIAL" : "G", "EMP_HIREDATE" : "1998-11-08", "JOB_CODE" : "5
02" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5b"), "EMP_NUM" : 105, "EMP_LNAME" : "Johnson"
, "EMP_FNAME" : "Alice", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1991-02-01", "JOB_CODE"
: "502" }
```

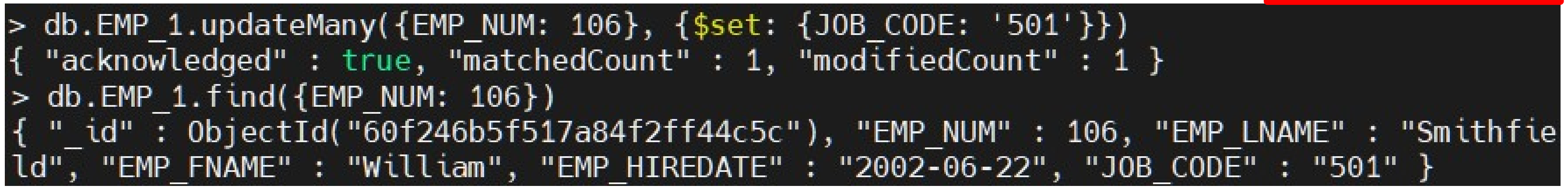
NoSQL Exercises

- Q4. Change the job code to 501 for the person whose personnel number is 106. After you have completed the task, examine the results, and reset the job code to its original value so that the database has not been changed.

NoSQL Exercises

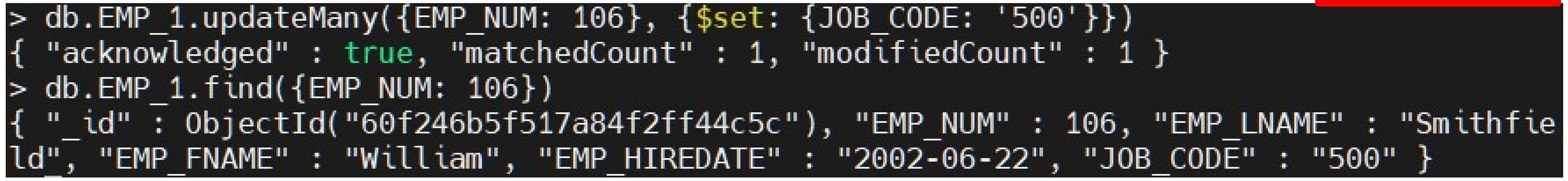
- A4.

Change the job code



```
> db.EMP_1.updateMany({EMP_NUM: 106}, {$set: {JOB_CODE: '501'}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.EMP_1.find({EMP_NUM: 106})
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5c"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfie
ld", "EMP_FNAME" : "William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "501" }
```

Restore the row



```
> db.EMP_1.updateMany({EMP_NUM: 106}, {$set: {JOB_CODE: '500'}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.EMP_1.find({EMP_NUM: 106})
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5c"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfie
ld", "EMP_FNAME" : "William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "500" }
```

(참조) oplog: DB의 시점 자체를 이전으로 돌리는 기능. DB 전체의 시점을 이동시켜 다른 컬렉션이나 사용자에게 영향을 끼치고 기능 자체의 cost가 크게 소모되어 사용을 지양

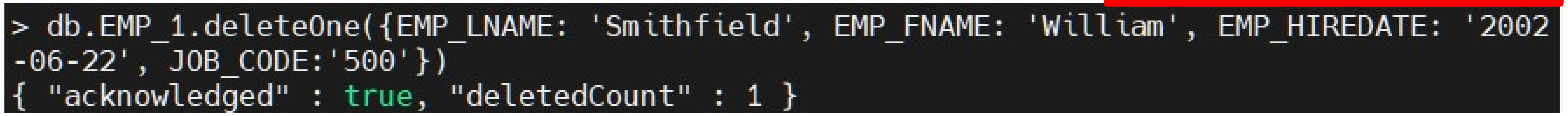
NoSQL Exercises

- Q5. Delete the row for the person named William Smithfield, who was hired on 2002-06-22 and whose job code classification is 500. After you have completed the task, examine the results, and restore the row so that the database has not been changed.

NoSQL Exercises

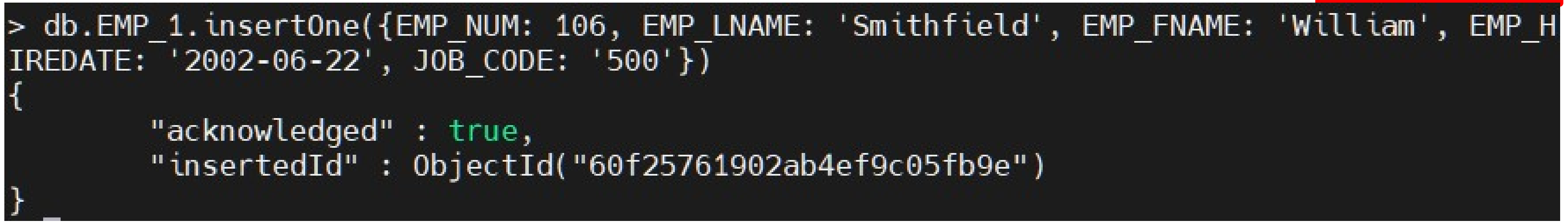
- A5.

Delete the the row for the person



```
> db.EMP_1.deleteOne({EMP_LNAME: 'Smithfield', EMP_FNAME: 'William', EMP_HIREDATE: '2002
-06-22', JOB_CODE:'500'})
{ "acknowledged" : true, "deletedCount" : 1 }
```

Restore the row



```
> db.EMP_1.insertOne({EMP_NUM: 106, EMP_LNAME: 'Smithfield', EMP_FNAME: 'William', EMP_H
IREDATE: '2002-06-22', JOB_CODE: '500'})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60f25761902ab4ef9c05fb9e")
}
```

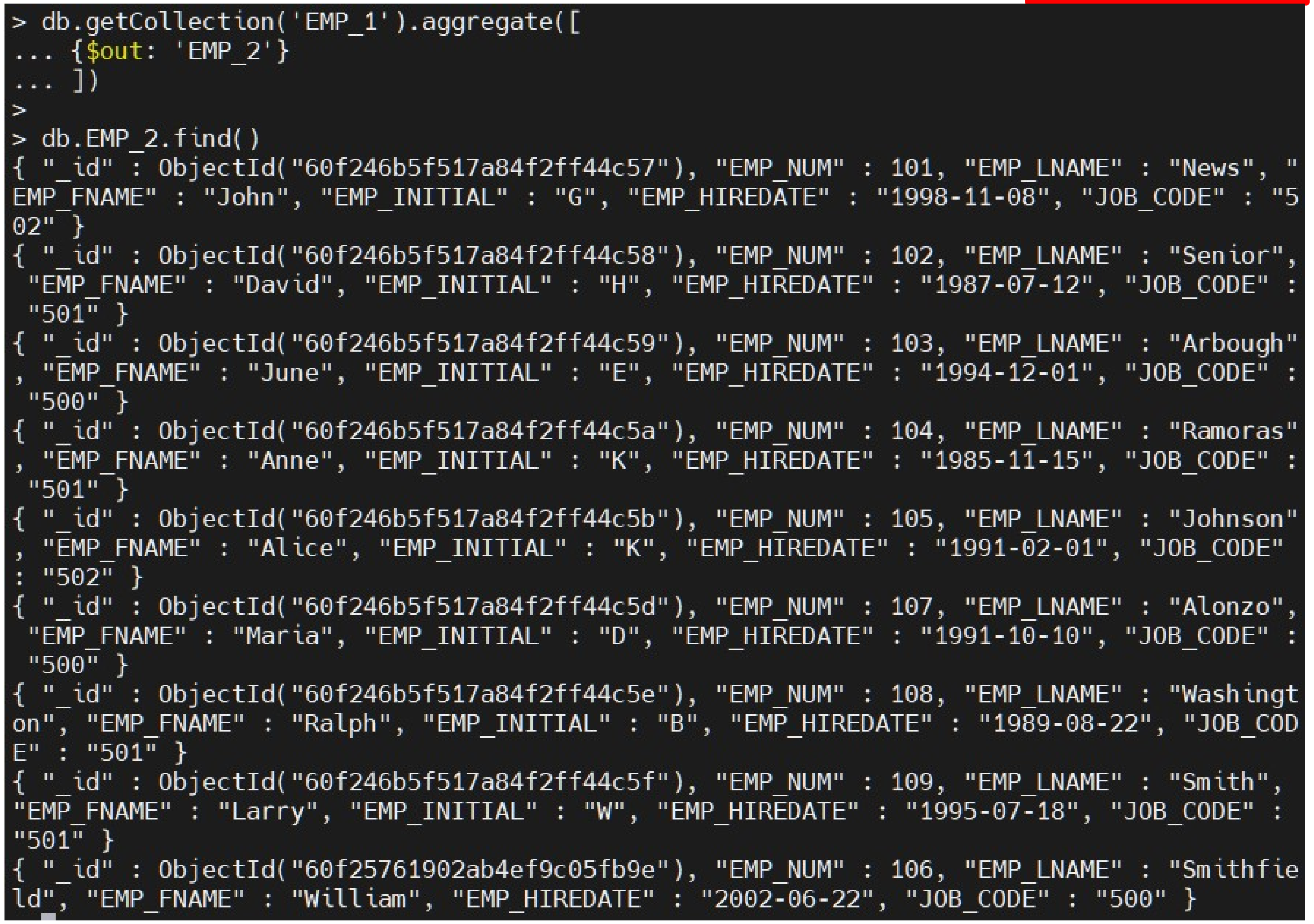
NoSQL Exercises

- Q6. Create a copy of EMP_1, naming the copy EMP_2. Then write the code that will add the attributes EMP_PCT and PROJ_NUM to its structure. The EMP_PCT is the bonus percentage to be paid to each employee.

NoSQL Exercises

- A6.

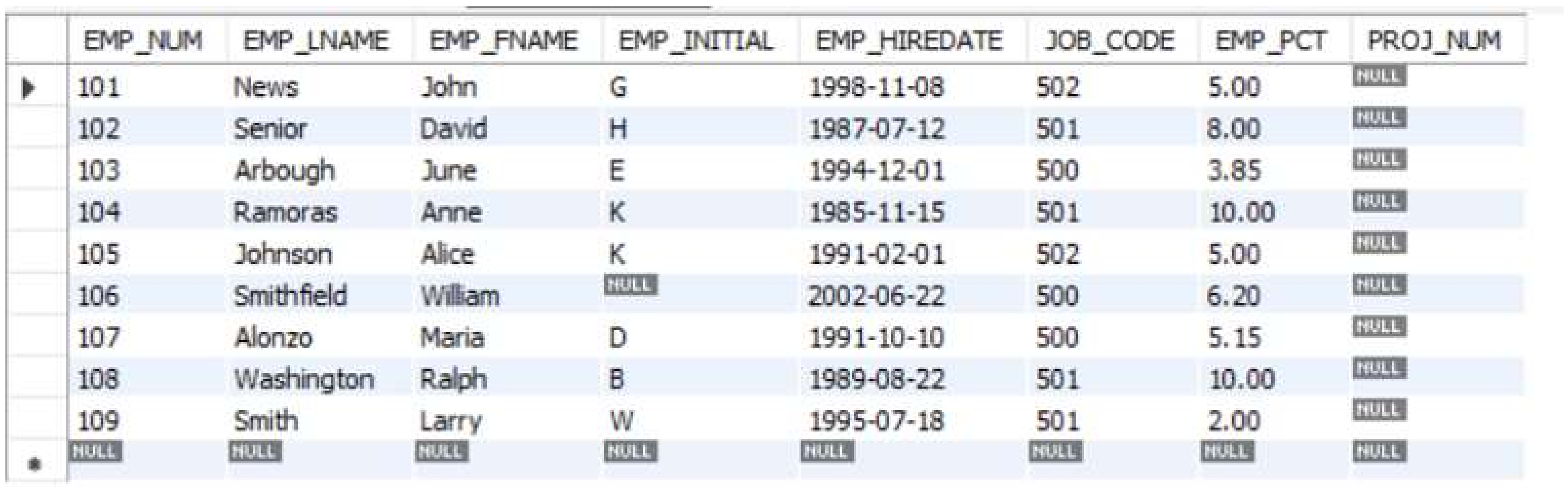
Copy the collection



```
> db.getCollection('EMP_1').aggregate([
... {$out: 'EMP_2'}
... ])
>
> db.EMP_2.find()
{ "_id" : ObjectId("60f246b5f517a84f2ff44c57"), "EMP_NUM" : 101, "EMP_LNAME" : "News", "
EMP_FNAME" : "John", "EMP_INITIAL" : "G", "EMP_HIREDATE" : "1998-11-08", "JOB_CODE" : "5
02" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c58"), "EMP_NUM" : 102, "EMP_LNAME" : "Senior",
 "EMP_FNAME" : "David", "EMP_INITIAL" : "H", "EMP_HIREDATE" : "1987-07-12", "JOB_CODE" :
 "501" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c59"), "EMP_NUM" : 103, "EMP_LNAME" : "Arbough"
, "EMP_FNAME" : "June", "EMP_INITIAL" : "E", "EMP_HIREDATE" : "1994-12-01", "JOB_CODE" :
 "500" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5a"), "EMP_NUM" : 104, "EMP_LNAME" : "Ramoras"
, "EMP_FNAME" : "Anne", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1985-11-15", "JOB_CODE" :
 "501" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5b"), "EMP_NUM" : 105, "EMP_LNAME" : "Johnson"
, "EMP_FNAME" : "Alice", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1991-02-01", "JOB_CODE"
: "502" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5d"), "EMP_NUM" : 107, "EMP_LNAME" : "Alonzo",
 "EMP_FNAME" : "Maria", "EMP_INITIAL" : "D", "EMP_HIREDATE" : "1991-10-10", "JOB_CODE" :
 "500" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5e"), "EMP_NUM" : 108, "EMP_LNAME" : "Washingt
on", "EMP_FNAME" : "Ralph", "EMP_INITIAL" : "B", "EMP_HIREDATE" : "1989-08-22", "JOB_COD
E" : "501" }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5f"), "EMP_NUM" : 109, "EMP_LNAME" : "Smith",
"EMP_FNAME" : "Larry", "EMP_INITIAL" : "W", "EMP_HIREDATE" : "1995-07-18", "JOB_CODE" :
"501" }
{ "_id" : ObjectId("60f25761902ab4ef9c05fb9e"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfie
ld", "EMP_FNAME" : "William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "500" }
```

NoSQL Exercises

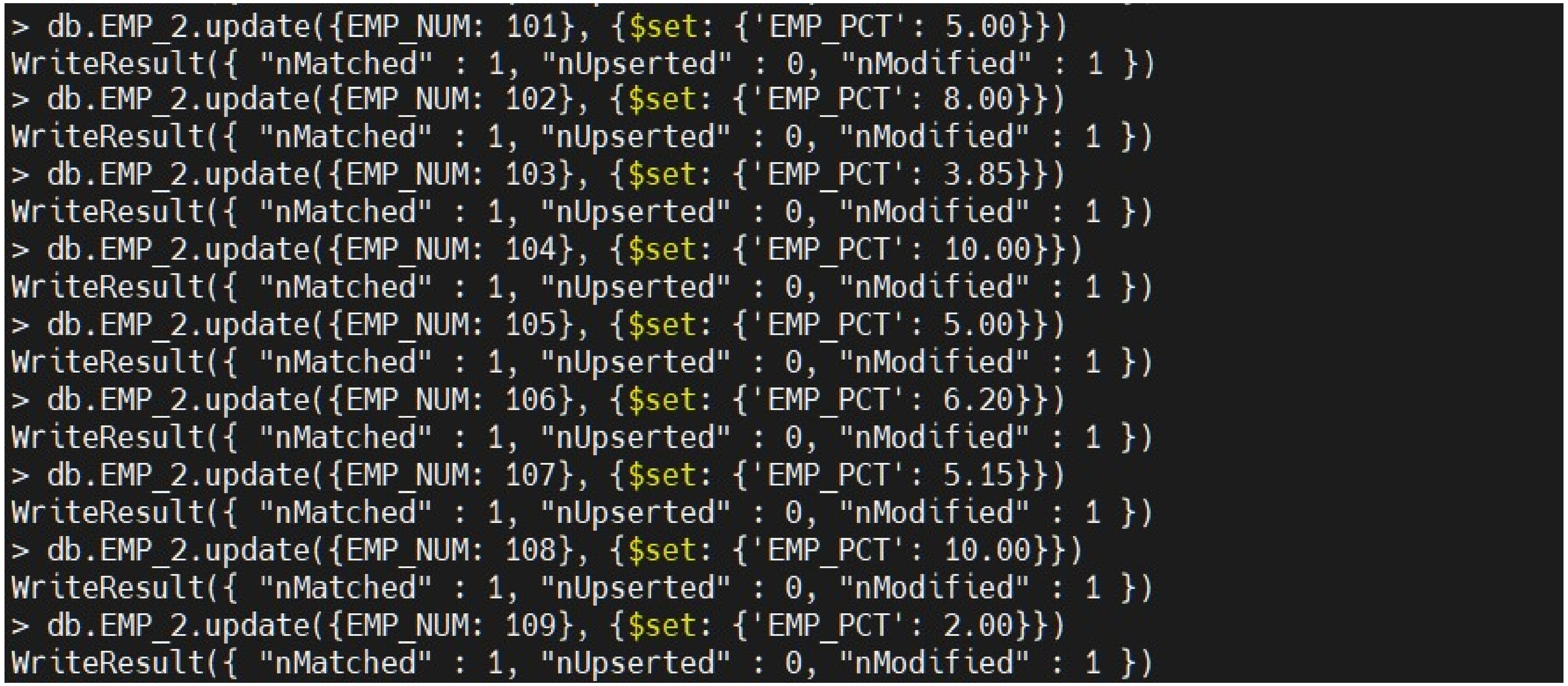
- Q7. Enter EMP_PCT value for each employee as below.



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE	EMP_PCT	PROJ_NUM
101	News	John	G	1998-11-08	502	5.00	NULL
102	Senior	David	H	1987-07-12	501	8.00	NULL
103	Arbough	June	E	1994-12-01	500	3.85	NULL
104	Ramoras	Anne	K	1985-11-15	501	10.00	NULL
105	Johnson	Alice	K	1991-02-01	502	5.00	NULL
106	Smithfield	William	NULL	2002-06-22	500	6.20	NULL
107	Alonzo	Maria	D	1991-10-10	500	5.15	NULL
108	Washington	Ralph	B	1989-08-22	501	10.00	NULL
109	Smith	Larry	W	1995-07-18	501	2.00	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

NoSQL Exercises

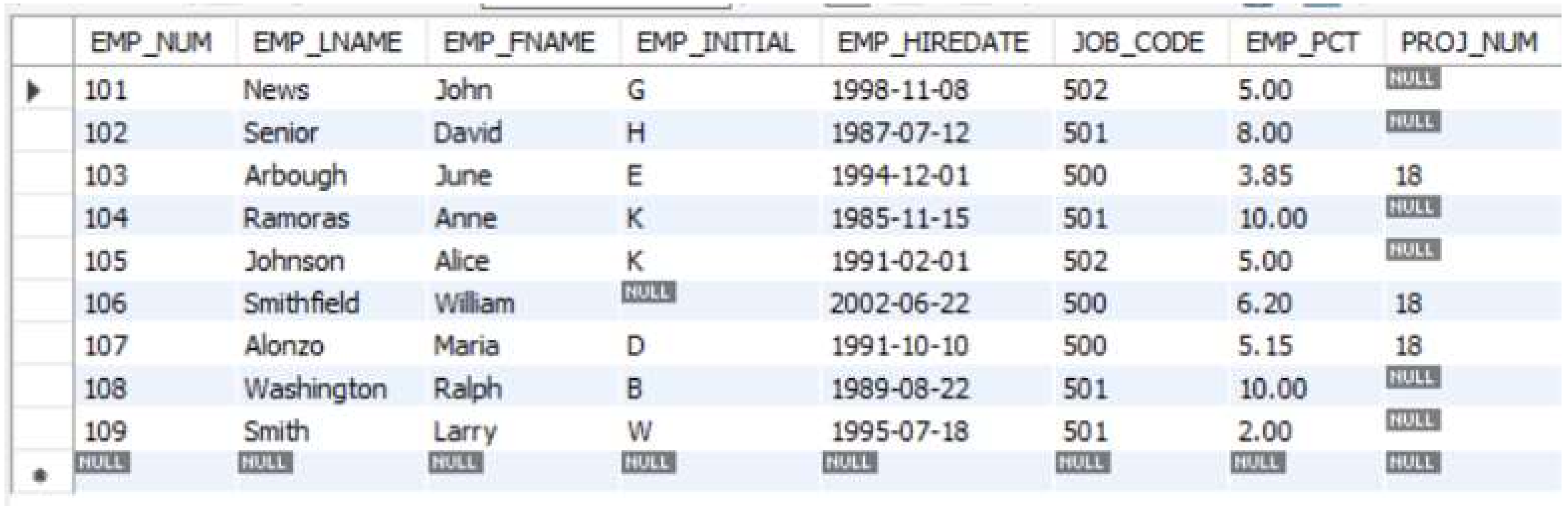
- A7.



```
> db.EMP_2.update({EMP_NUM: 101}, {$set: {'EMP_PCT': 5.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 102}, {$set: {'EMP_PCT': 8.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 103}, {$set: {'EMP_PCT': 3.85}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 104}, {$set: {'EMP_PCT': 10.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 105}, {$set: {'EMP_PCT': 5.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 106}, {$set: {'EMP_PCT': 6.20}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 107}, {$set: {'EMP_PCT': 5.15}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 108}, {$set: {'EMP_PCT': 10.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.EMP_2.update({EMP_NUM: 109}, {$set: {'EMP_PCT': 2.00}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

NoSQL Exercises

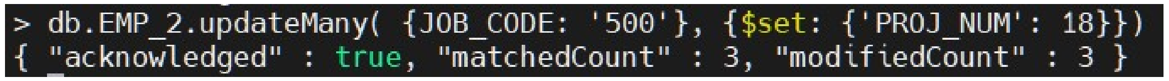
- Q8. Using a single command sequence, write the code that will enter the project number (PROJ_NUM) = 18 for all employees whose job classification (JOB_CODE) is 500.



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE	EMP_PCT	PROJ_NUM
101	News	John	G	1998-11-08	502	5.00	NULL
102	Senior	David	H	1987-07-12	501	8.00	NULL
103	Arbough	June	E	1994-12-01	500	3.85	18
104	Ramoras	Anne	K	1985-11-15	501	10.00	NULL
105	Johnson	Alice	K	1991-02-01	502	5.00	NULL
106	Smithfield	William	NULL	2002-06-22	500	6.20	18
107	Alonzo	Maria	D	1991-10-10	500	5.15	18
108	Washington	Ralph	B	1989-08-22	501	10.00	NULL
109	Smith	Larry	W	1995-07-18	501	2.00	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

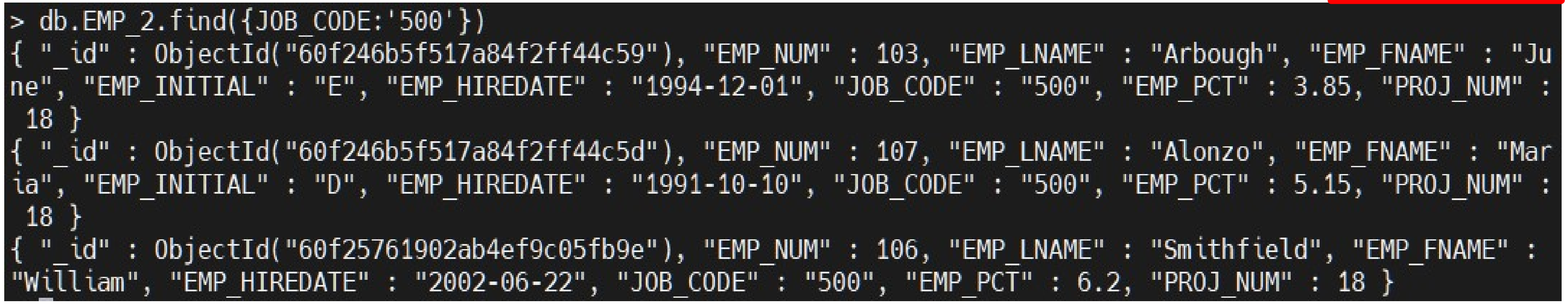
NoSQL Exercises

- A8.



```
> db.EMP_2.updateMany( {JOB_CODE: '500'}, {$set: {'PROJ_NUM': 18}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

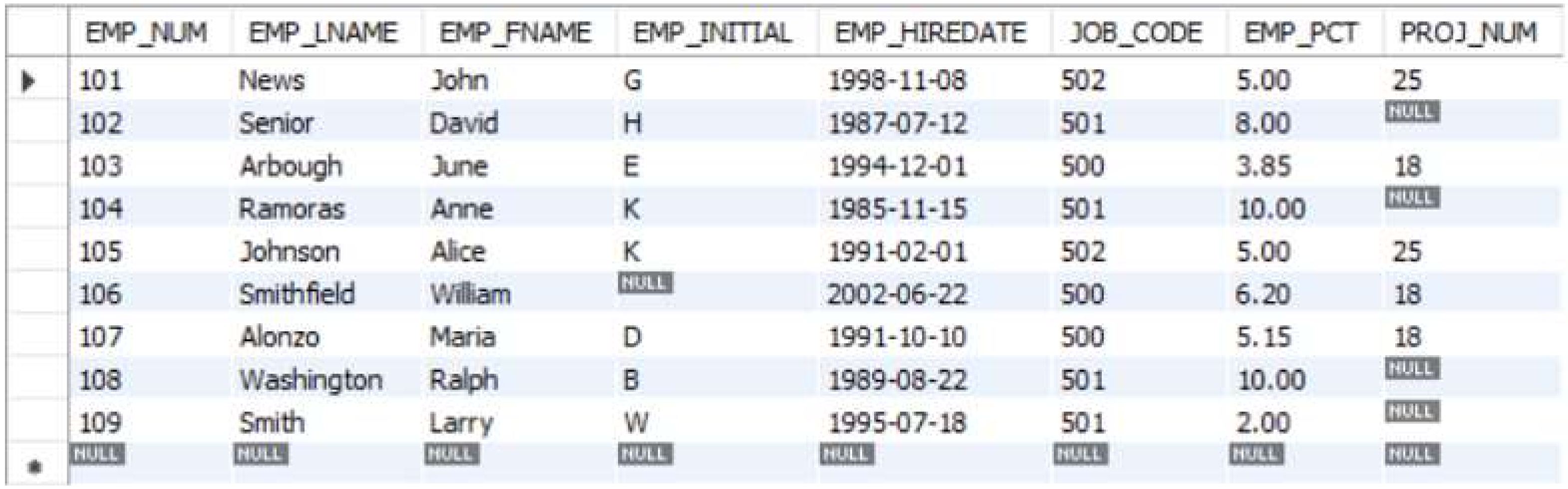
Check the result



```
> db.EMP_2.find({JOB_CODE:'500'})
{ "_id" : ObjectId("60f246b5f517a84f2ff44c59"), "EMP_NUM" : 103, "EMP_LNAME" : "Arbough", "EMP_FNAME" : "Ju
ne", "EMP_INITIAL" : "E", "EMP_HIREDATE" : "1994-12-01", "JOB_CODE" : "500", "EMP_PCT" : 3.85, "PROJ_NUM" :
 18 }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5d"), "EMP_NUM" : 107, "EMP_LNAME" : "Alonzo", "EMP_FNAME" : "Mar
ia", "EMP_INITIAL" : "D", "EMP_HIREDATE" : "1991-10-10", "JOB_CODE" : "500", "EMP_PCT" : 5.15, "PROJ_NUM" :
 18 }
{ "_id" : ObjectId("60f25761902ab4ef9c05fb9e"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfield", "EMP_FNAME" :
"William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "500", "EMP_PCT" : 6.2, "PROJ_NUM" : 18 }
```

NoSQL Exercises

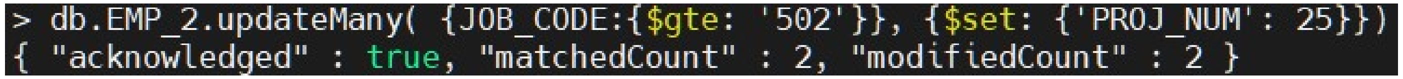
- Q9. Using a single command sequence, write the code that will enter the project number (PROJ_NUM) = 25 for all employees whose job classification (JOB_CODE) is 502 or higher. When you are done with questions 8 and 9, the EMP_2 table will contain the data as below.



	EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE	EMP_PCT	PROJ_NUM
▶	101	News	John	G	1998-11-08	502	5.00	25
	102	Senior	David	H	1987-07-12	501	8.00	NULL
	103	Arbough	June	E	1994-12-01	500	3.85	18
	104	Ramoras	Anne	K	1985-11-15	501	10.00	NULL
	105	Johnson	Alice	K	1991-02-01	502	5.00	25
	106	Smithfield	William	NULL	2002-06-22	500	6.20	18
	107	Alonzo	Maria	D	1991-10-10	500	5.15	18
	108	Washington	Ralph	B	1989-08-22	501	10.00	NULL
	109	Smith	Larry	W	1995-07-18	501	2.00	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

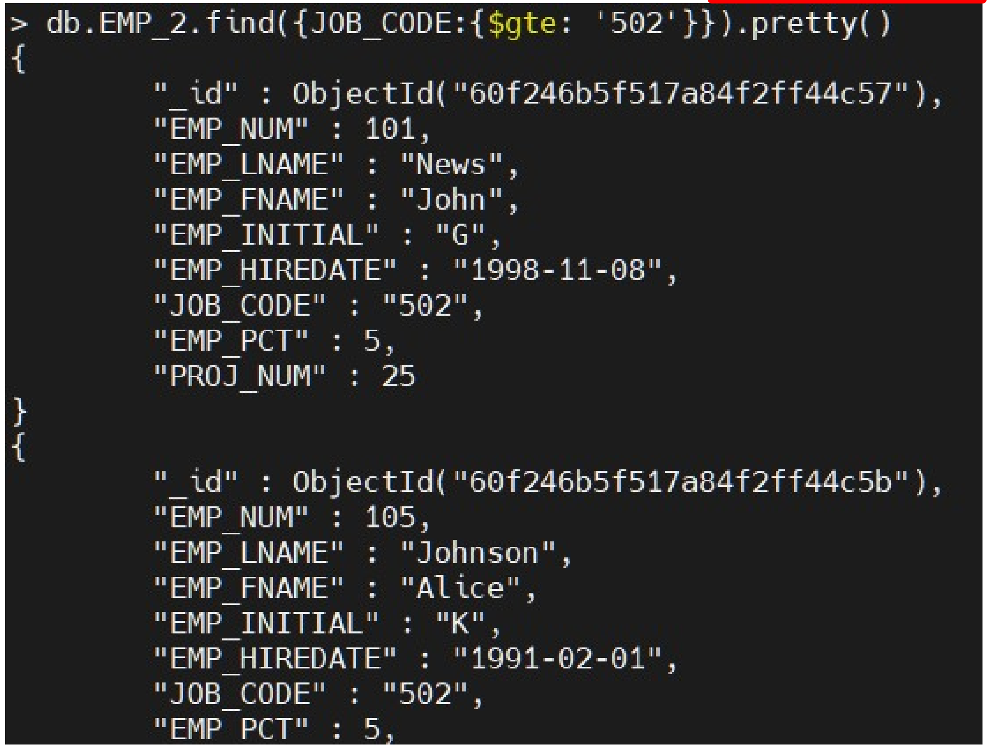
NoSQL Exercises

- A9.



```
> db.EMP_2.updateMany( {JOB_CODE:{$gte: '502'}}, {$set: {'PROJ_NUM': 25}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
```

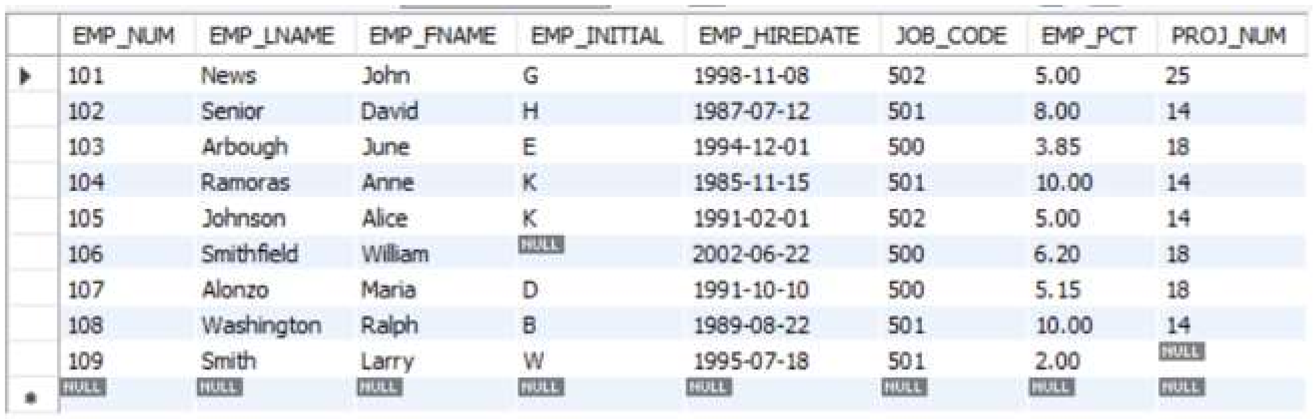
Check the result



```
> db.EMP_2.find({JOB_CODE:{$gte: '502'}}).pretty()
{
        "_id" : ObjectId("60f246b5f517a84f2ff44c57"),
        "EMP_NUM" : 101,
        "EMP_LNAME" : "News",
        "EMP_FNAME" : "John",
        "EMP_INITIAL" : "G",
        "EMP_HIREDATE" : "1998-11-08",
        "JOB_CODE" : "502",
        "EMP_PCT" : 5,
        "PROJ_NUM" : 25
}
{
        "_id" : ObjectId("60f246b5f517a84f2ff44c5b"),
        "EMP_NUM" : 105,
        "EMP_LNAME" : "Johnson",
        "EMP_FNAME" : "Alice",
        "EMP_INITIAL" : "K",
        "EMP_HIREDATE" : "1991-02-01",
        "JOB_CODE" : "502",
        "EMP_PCT" : 5,
```

NoSQL Exercises

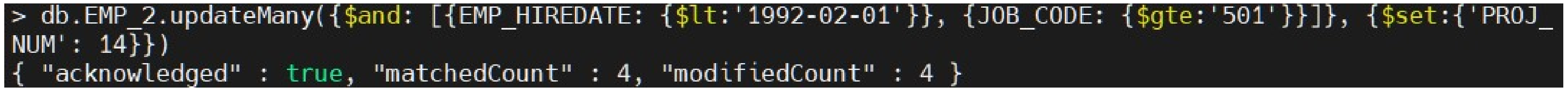
- Q10. Write the code that will enter a PROJ_NUM of 14 for those employees who were hired before January 1, 1992 and whose job code is at least 501.



	EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE	EMP_PCT	PROJ_NUM
▶	101	News	John	G	1998-11-08	502	5.00	25
	102	Senior	David	H	1987-07-12	501	8.00	14
	103	Arbough	June	E	1994-12-01	500	3.85	18
	104	Ramoras	Anne	K	1985-11-15	501	10.00	14
	105	Johnson	Alice	K	1991-02-01	502	5.00	14
	106	Smithfield	William	NULL	2002-06-22	500	6.20	18
	107	Alonzo	Maria	D	1991-10-10	500	5.15	18
	108	Washington	Ralph	B	1989-08-22	501	10.00	14
	109	Smith	Larry	W	1995-07-18	501	2.00	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

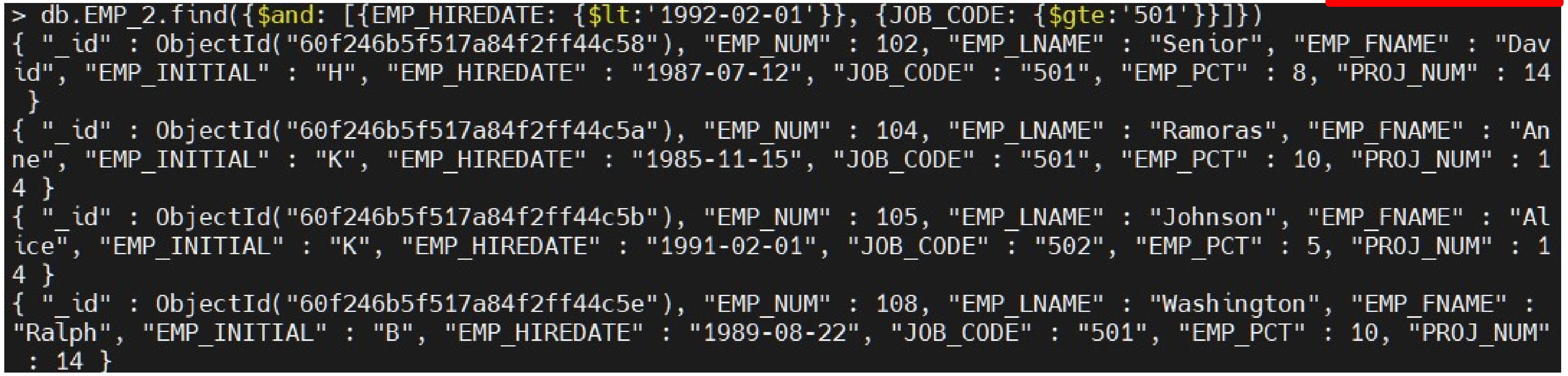
NoSQL Exercises

- A10.



```
> db.EMP_2.updateMany({$and: [{EMP_HIREDATE: {$lt:'1992-02-01'}}, {JOB_CODE: {$gte:'501'}}]}, {$set:{'PROJ_
NUM': 14}})
{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 4 }
```

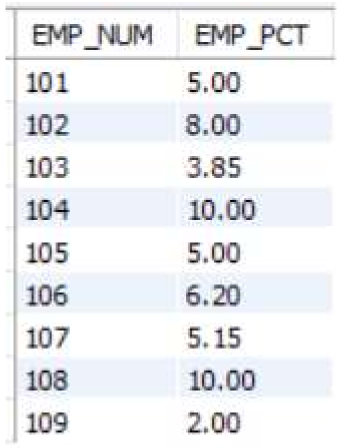
Check the result



```
> db.EMP_2.find({$and: [{EMP_HIREDATE: {$lt:'1992-02-01'}}, {JOB_CODE: {$gte:'501'}}]})
{ "_id" : ObjectId("60f246b5f517a84f2ff44c58"), "EMP_NUM" : 102, "EMP_LNAME" : "Senior", "EMP_FNAME" : "Dav
id", "EMP_INITIAL" : "H", "EMP_HIREDATE" : "1987-07-12", "JOB_CODE" : "501", "EMP_PCT" : 8, "PROJ_NUM" : 14
 }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5a"), "EMP_NUM" : 104, "EMP_LNAME" : "Ramoras", "EMP_FNAME" : "An
ne", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1985-11-15", "JOB_CODE" : "501", "EMP_PCT" : 10, "PROJ_NUM" : 1
4 }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5b"), "EMP_NUM" : 105, "EMP_LNAME" : "Johnson", "EMP_FNAME" : "Al
ice", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1991-02-01", "JOB_CODE" : "502", "EMP_PCT" : 5, "PROJ_NUM" : 1
4 }
{ "_id" : ObjectId("60f246b5f517a84f2ff44c5e"), "EMP_NUM" : 108, "EMP_LNAME" : "Washington", "EMP_FNAME" :
"Ralph", "EMP_INITIAL" : "B", "EMP_HIREDATE" : "1989-08-22", "JOB_CODE" : "501", "EMP_PCT" : 10, "PROJ_NUM"
 : 14 }
```

NoSQL Exercises

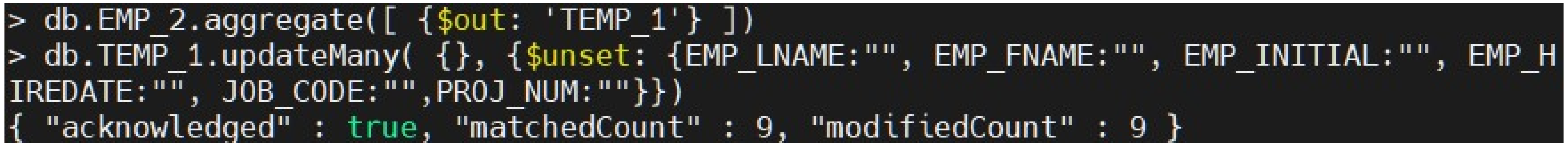
- Q11. Create a temporary table named TEMP_1, whose structure is composed of the EMP_2 attributes EMP_NUM and EMP_PCT. Copy the matching EMP_2 values into the TEMP_1 table.



EMP_NUM	EMP_PCT
101	5.00
102	8.00
103	3.85
104	10.00
105	5.00
106	6.20
107	5.15
108	10.00
109	2.00

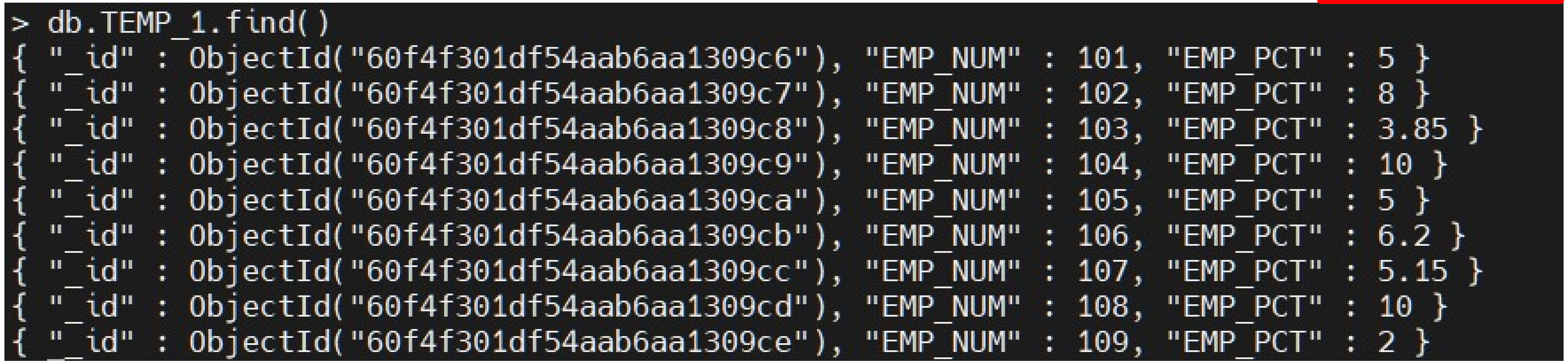
NoSQL Exercises

- A11.



```
> db.EMP_2.aggregate([ {$out: 'TEMP_1'} ])
> db.TEMP_1.updateMany( {}, {$unset: {EMP_LNAME:"", EMP_FNAME:"", EMP_INITIAL:"", EMP_H
IREDATE:"", JOB_CODE:"",PROJ_NUM:""}})
{ "acknowledged" : true, "matchedCount" : 9, "modifiedCount" : 9 }
```

Check the result



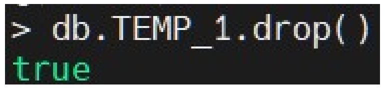
```
> db.TEMP_1.find()
{ "_id" : ObjectId("60f4f301df54aab6aa1309c6"), "EMP_NUM" : 101, "EMP_PCT" : 5 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c7"), "EMP_NUM" : 102, "EMP_PCT" : 8 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c8"), "EMP_NUM" : 103, "EMP_PCT" : 3.85 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c9"), "EMP_NUM" : 104, "EMP_PCT" : 10 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309ca"), "EMP_NUM" : 105, "EMP_PCT" : 5 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cb"), "EMP_NUM" : 106, "EMP_PCT" : 6.2 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cc"), "EMP_NUM" : 107, "EMP_PCT" : 5.15 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cd"), "EMP_NUM" : 108, "EMP_PCT" : 10 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309ce"), "EMP_NUM" : 109, "EMP_PCT" : 2 }
```

NoSQL Exercises

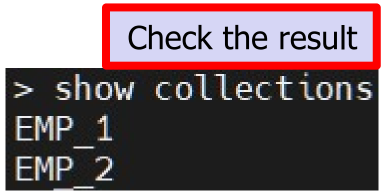
- Q12. Write the SQL command that will delete the newly created TEMP_1 table from the database.

NoSQL Exercises

- A12.



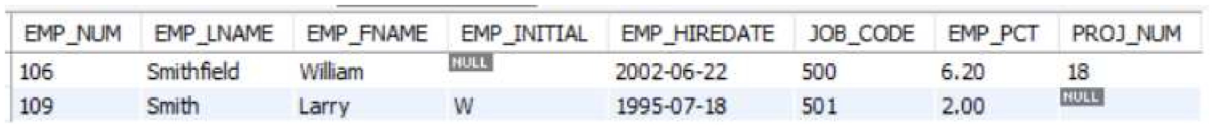
```
> db.TEMP_1.drop()
true
```



```
> show collections
EMP_1
EMP_2
```

NoSQL Exercises

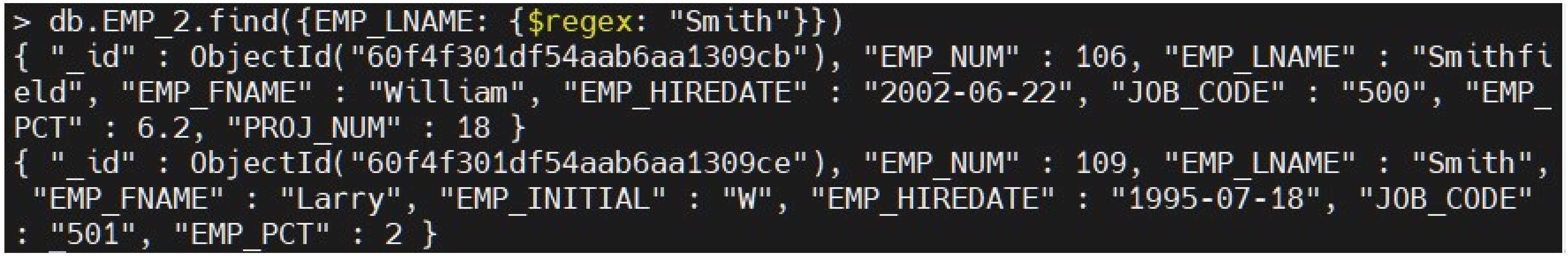
- Q13. Write the code required to list all employees whose last names start with 'Smith'. In other words, the rows for both Smith and Smithfield should be included in the listing.



EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE	EMP_PCT	PROJ_NUM
106	Smithfield	William	NULL	2002-06-22	500	6.20	18
109	Smith	Larry	W	1995-07-18	501	2.00	NULL

NoSQL Exercises

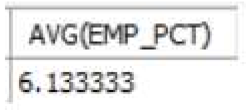
- A13.



```
> db.EMP_2.find({EMP_LNAME: {$regex: "Smith"}})
{ "_id" : ObjectId("60f4f301df54aab6aa1309cb"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfi
eld", "EMP_FNAME" : "William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "500", "EMP_
PCT" : 6.2, "PROJ_NUM" : 18 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309ce"), "EMP_NUM" : 109, "EMP_LNAME" : "Smith",
 "EMP_FNAME" : "Larry", "EMP_INITIAL" : "W", "EMP_HIREDATE" : "1995-07-18", "JOB_CODE"
: "501", "EMP_PCT" : 2 }
```

NoSQL Exercises

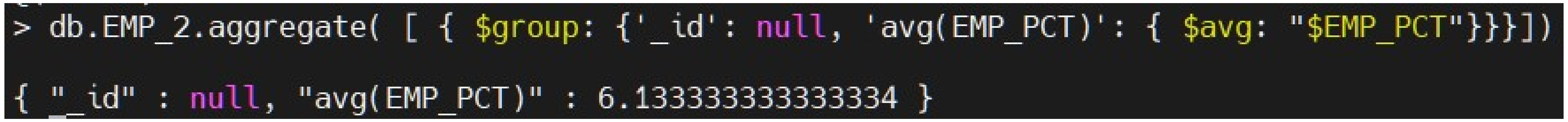
- Q17. Write the code to find the average bonus percentage in the EMP_2 table you created in question 6.



```
AVG(EMP_PCT)
6.133333
```

NoSQL Exercises

- A17.



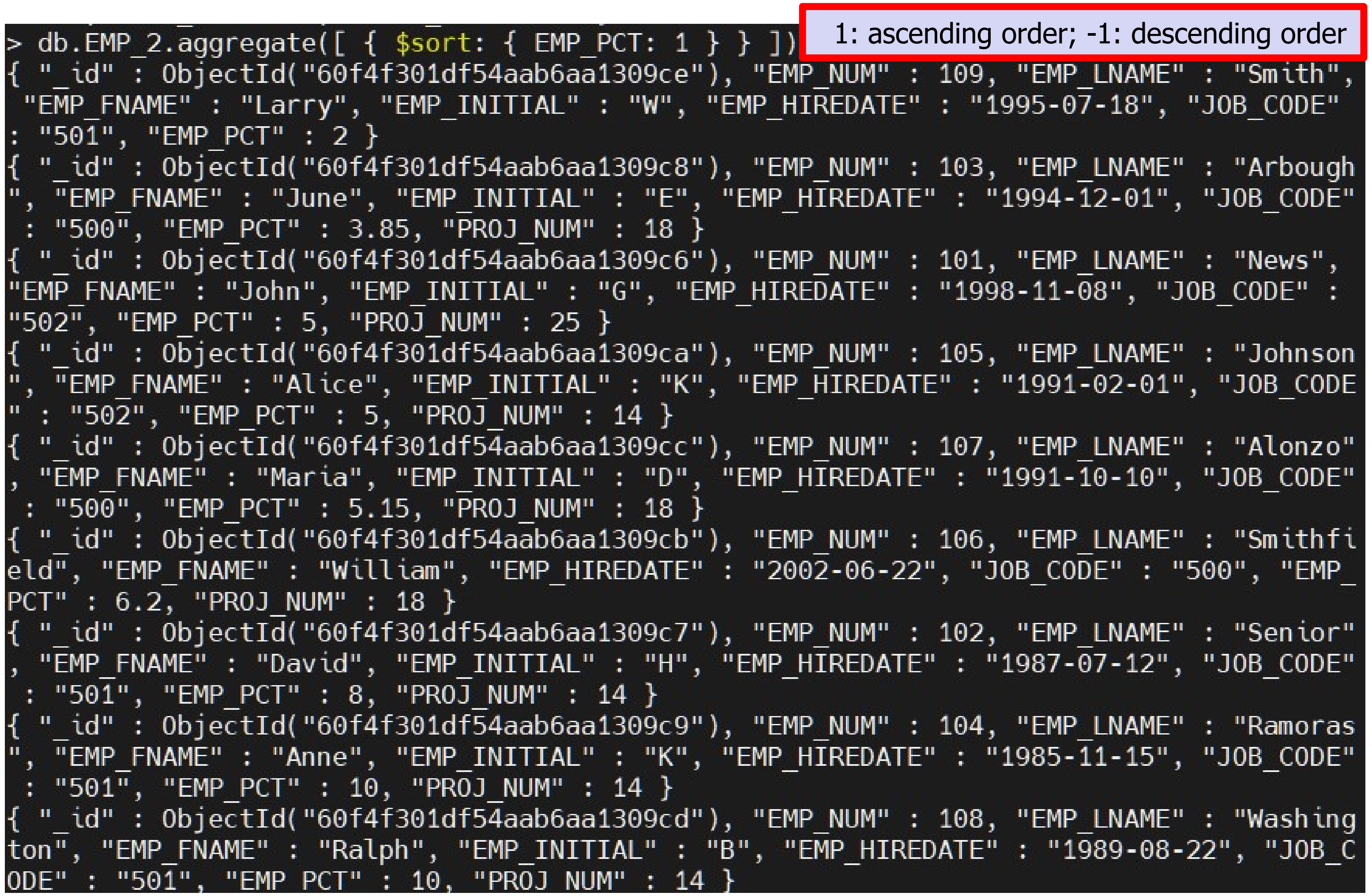
```
> db.EMP_2.aggregate( [ { $group: {'_id': null, 'avg(EMP_PCT)': { $avg: "$EMP_PCT"}}}])

{ "_id" : null, "avg(EMP_PCT)" : 6.133333333333334 }
```

NoSQL Exercises

- Q18. Write the code that will produce a listing for the data in the EMP_2 table in ascending order by the bonus percentage.

NoSQL Exercises

- A18.



1: ascending order; -1: descending order

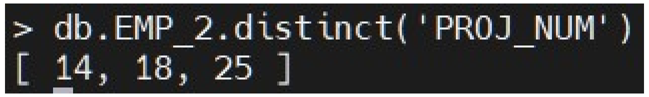
```
> db.EMP_2.aggregate([ { $sort: { EMP_PCT: 1 } } ])
{ "_id" : ObjectId("60f4f301df54aab6aa1309ce"), "EMP_NUM" : 109, "EMP_LNAME" : "Smith",
 "EMP_FNAME" : "Larry", "EMP_INITIAL" : "W", "EMP_HIREDATE" : "1995-07-18", "JOB_CODE"
: "501", "EMP_PCT" : 2 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c8"), "EMP_NUM" : 103, "EMP_LNAME" : "Arbough
", "EMP_FNAME" : "June", "EMP_INITIAL" : "E", "EMP_HIREDATE" : "1994-12-01", "JOB_CODE"
 : "500", "EMP_PCT" : 3.85, "PROJ_NUM" : 18 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c6"), "EMP_NUM" : 101, "EMP_LNAME" : "News",
"EMP_FNAME" : "John", "EMP_INITIAL" : "G", "EMP_HIREDATE" : "1998-11-08", "JOB_CODE" :
"502", "EMP_PCT" : 5, "PROJ_NUM" : 25 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309ca"), "EMP_NUM" : 105, "EMP_LNAME" : "Johnson
", "EMP_FNAME" : "Alice", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1991-02-01", "JOB_CODE
" : "502", "EMP_PCT" : 5, "PROJ_NUM" : 14 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cc"), "EMP_NUM" : 107, "EMP_LNAME" : "Alonzo"
, "EMP_FNAME" : "Maria", "EMP_INITIAL" : "D", "EMP_HIREDATE" : "1991-10-10", "JOB_CODE"
 : "500", "EMP_PCT" : 5.15, "PROJ_NUM" : 18 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cb"), "EMP_NUM" : 106, "EMP_LNAME" : "Smithfi
eld", "EMP_FNAME" : "William", "EMP_HIREDATE" : "2002-06-22", "JOB_CODE" : "500", "EMP_
PCT" : 6.2, "PROJ_NUM" : 18 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c7"), "EMP_NUM" : 102, "EMP_LNAME" : "Senior"
, "EMP_FNAME" : "David", "EMP_INITIAL" : "H", "EMP_HIREDATE" : "1987-07-12", "JOB_CODE"
 : "501", "EMP_PCT" : 8, "PROJ_NUM" : 14 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309c9"), "EMP_NUM" : 104, "EMP_LNAME" : "Ramoras
", "EMP_FNAME" : "Anne", "EMP_INITIAL" : "K", "EMP_HIREDATE" : "1985-11-15", "JOB_CODE"
 : "501", "EMP_PCT" : 10, "PROJ_NUM" : 14 }
{ "_id" : ObjectId("60f4f301df54aab6aa1309cd"), "EMP_NUM" : 108, "EMP_LNAME" : "Washing
ton", "EMP_FNAME" : "Ralph", "EMP_INITIAL" : "B", "EMP_HIREDATE" : "1989-08-22", "JOB_C
ODE" : "501", "EMP_PCT" : 10, "PROJ_NUM" : 14 }
```

NoSQL Exercises

- Q19. Write the SQL code that will list only the different project numbers found in the EMP_2 table.

NoSQL Exercises

- A19.



```
> db.EMP_2.distinct('PROJ_NUM')
[ 14, 18, 25 ]
```