No SQL

Sql vs No sql

language

approach

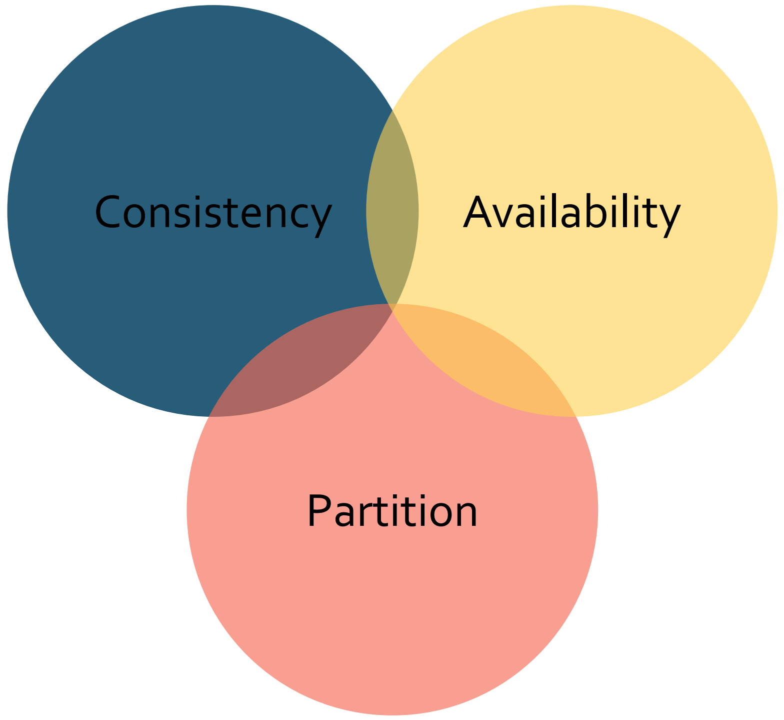
RDBMS	NoSQL
Rigid	flexible
Data stored in tables	Data can be stored in : tables, json objects, graphs, nodes and edges
Reduces data duplication	Enables scaling and rapid application changes

TECHNION
Azrieli Continuing Education and External Studies Division

CAP theorem

Distributed database can guarantee only **two** of the following three properties at the same time:

1. **Consistency (C)**: All nodes see the same data at the same time.
2. **Availability (A)**: Every request gets a response, even if some nodes are down.
3. **Partition Tolerance (P)**: The system continues to function despite network failures splitting communication between nodes.



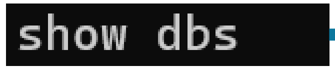
Mongo- about

Origin of name

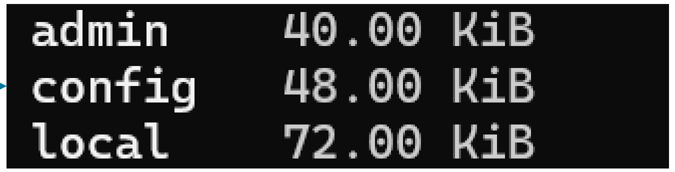
SQL	MongoDB
Data Base	Data Base
Table	Collection
Row	Document
column	field

MongoDB shell Commands

show dbs– to display the databases

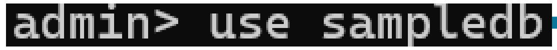


```
show dbs
```

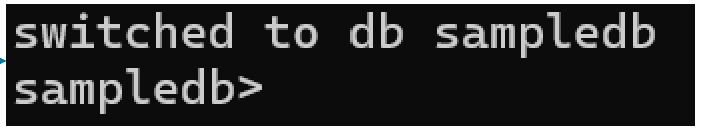


```
admin     40.00 KiB
config    48.00 KiB
local     72.00 KiB
```

use – to use a database or to create one

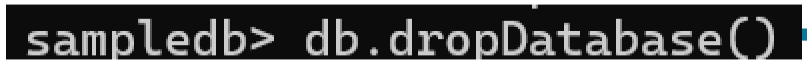


```
admin> use sampledb
```

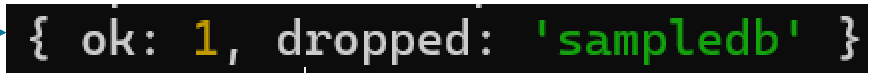


```
switched to db sampledb
sampledb>
```

drop to remove the database completely



```
sampledb> db.dropDatabase()
```



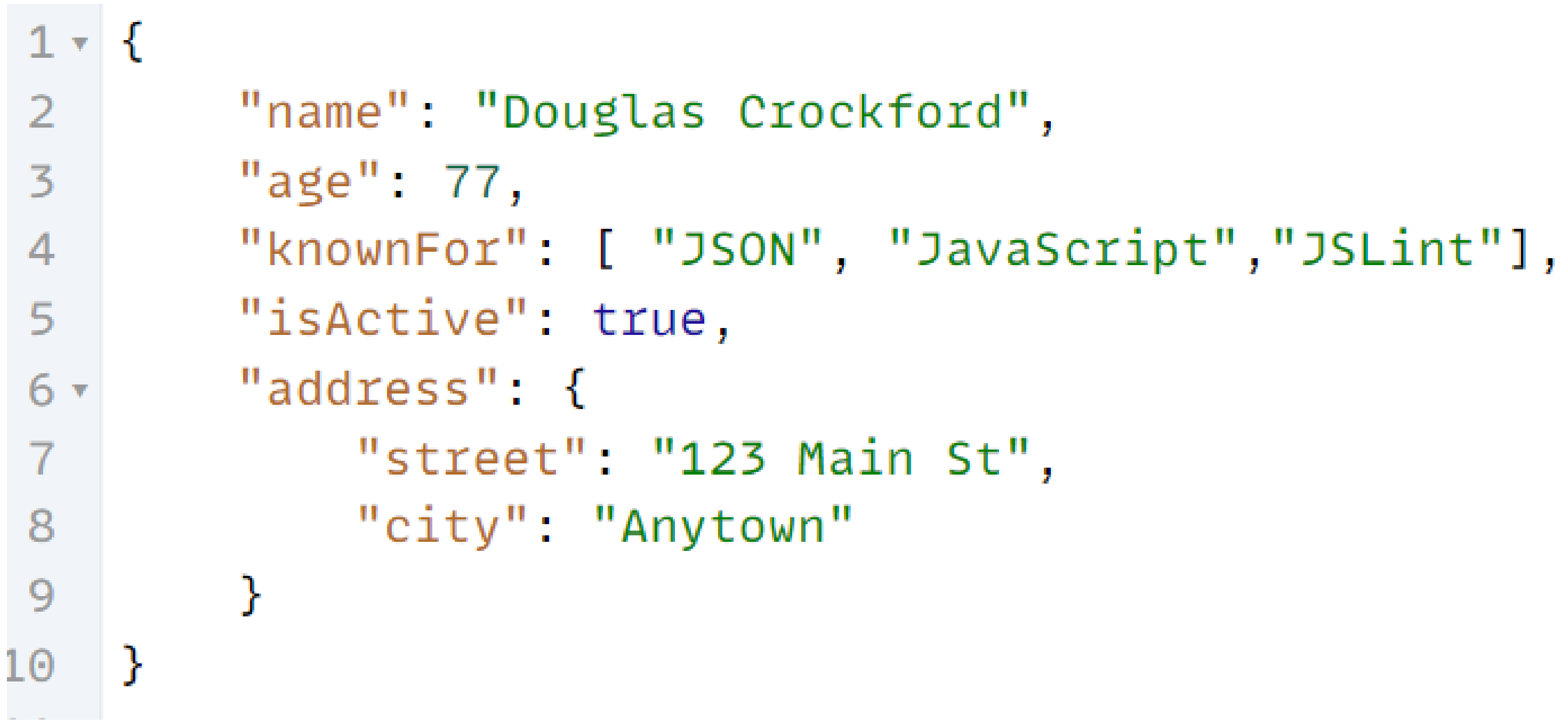
```
{ ok: 1, dropped: 'sampledb' }
```

JSON - **JavaScript Object Notation.**

- **Concept**:
  - JSON is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
  - It is often used to exchange data between a server and a web application.
  - **Syntax**: It consists of key-value pairs, similar to objects in programming languages.
- **History**:
  - JSON was originally derived from JavaScript in the early 2000s.
  - It became popular due to its simplicity and the rise of JavaScript-based web applications (AJAX).
  - JSON was formalized by Douglas Crockford in 2001 and is now a standard format used globally.
- **Need**:
  - JSON allows for easy and efficient data exchange between systems.
  - It is language-independent but can be parsed by most modern programming languages, making it ideal for web APIs and services.

TECHNION
Azrieli Continuing Education and External Studies Division

JSON Format



```
{
    "name": "Douglas Crockford",
    "age": 77,
    "knownFor": [ "JSON", "JavaScript","JSLint"],
    "isActive": true,
    "address": {
        "street": "123 Main St",
        "city": "Anytown"
    }
}
```

- **Key-Value Pairs**: Each pair is a string (key) and a corresponding value.
- **Arrays**: JSON supports arrays for ordered collections (e.g., "courses": ["JSON", "JavaScript","JSLint"]).
- **Nested Objects**: You can embed objects within objects (e.g., "address").

JSON Format

Advantages of JSON

- **Human-readable**: Easy to understand and edit.
- **Lightweight**: Compact, with minimal overhead.
- **Language-independent**: Supported by nearly all programming languages.
- **Interoperability**: Standard format for APIs (e.g., REST APIs).
- **Easy parsing**: Fast to parse and generate by machines.

Links to JSON Editors

- JSONLint - JSON validator and formatter.
- JSON Editor Online - Visual tool for editing JSON data.
- jsonformatter.org - Formatter, validator, and beautifier for JSON.

Mongosh Commands

Creating Collections

explicit

By running
a create
command

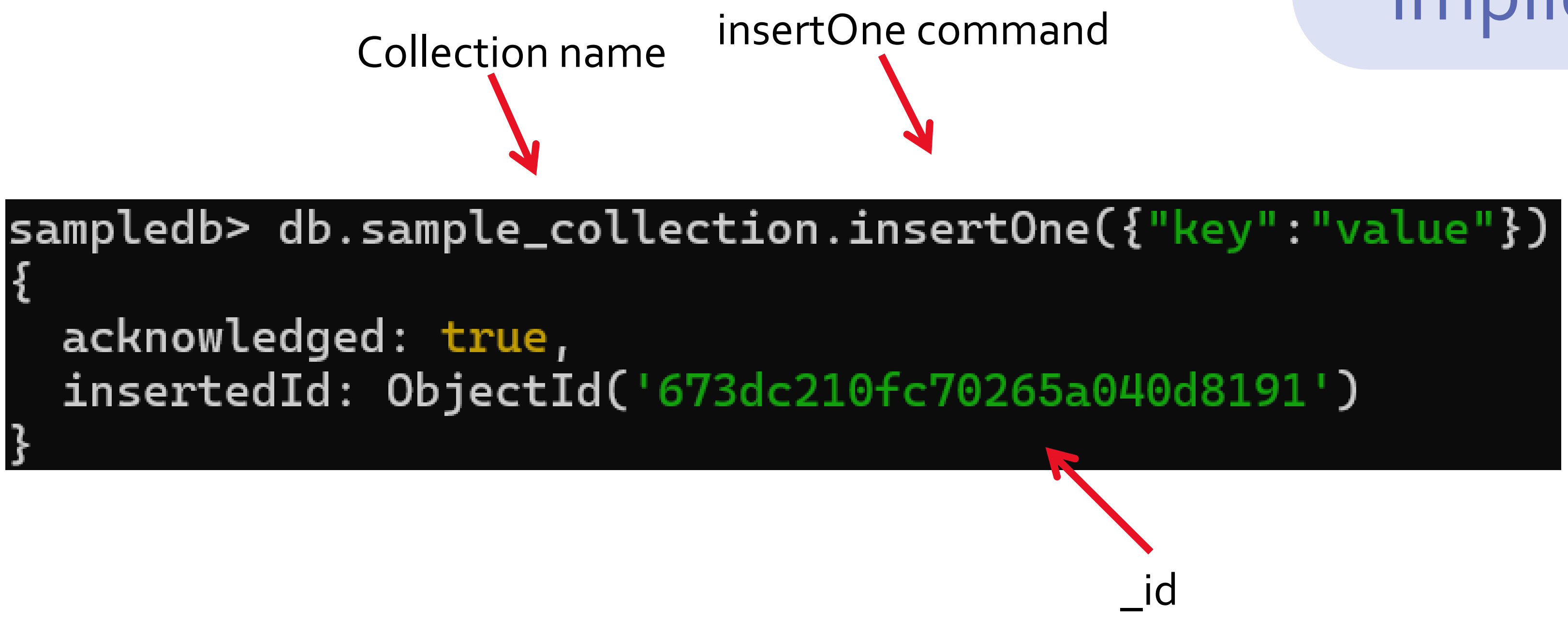
implicit

By
inserting
a row

Mongosh Commands

Creating Collections

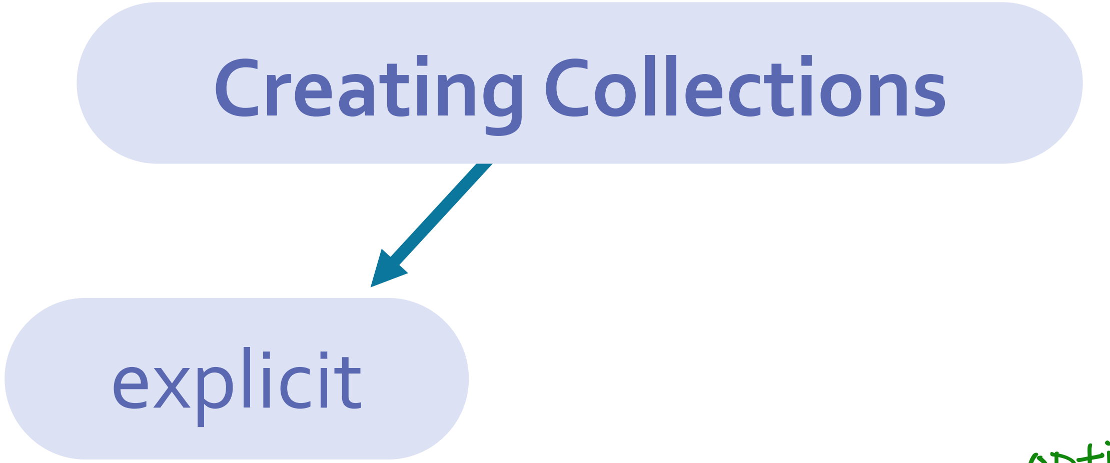
implicit

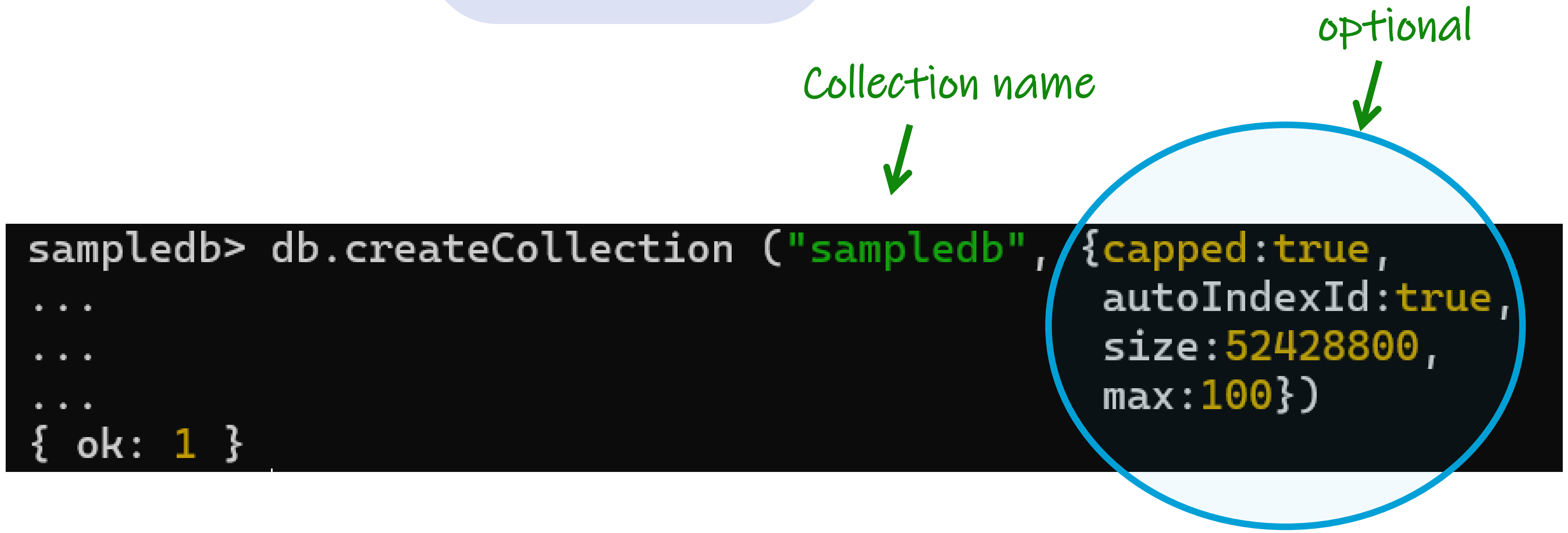


```
sampledb> db.sample_collection.insertOne({"key":"value"})
{
  acknowledged: true,
  insertedId: ObjectId('673dc210fc70265a040d8191')
}
```

TECHNION
Azrieli Continuing Education and External Studies Division

Mongosh Commands

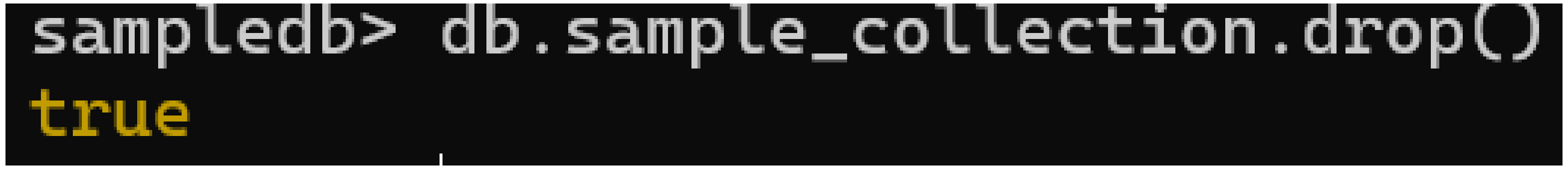




```
sampledb> db.createCollection ("sampledb", {capped:true,
...                                          autoIndexId:true,
...                                          size:52428800,
...                                          max:100})
{ ok: 1 }
```

Mongosh Commands

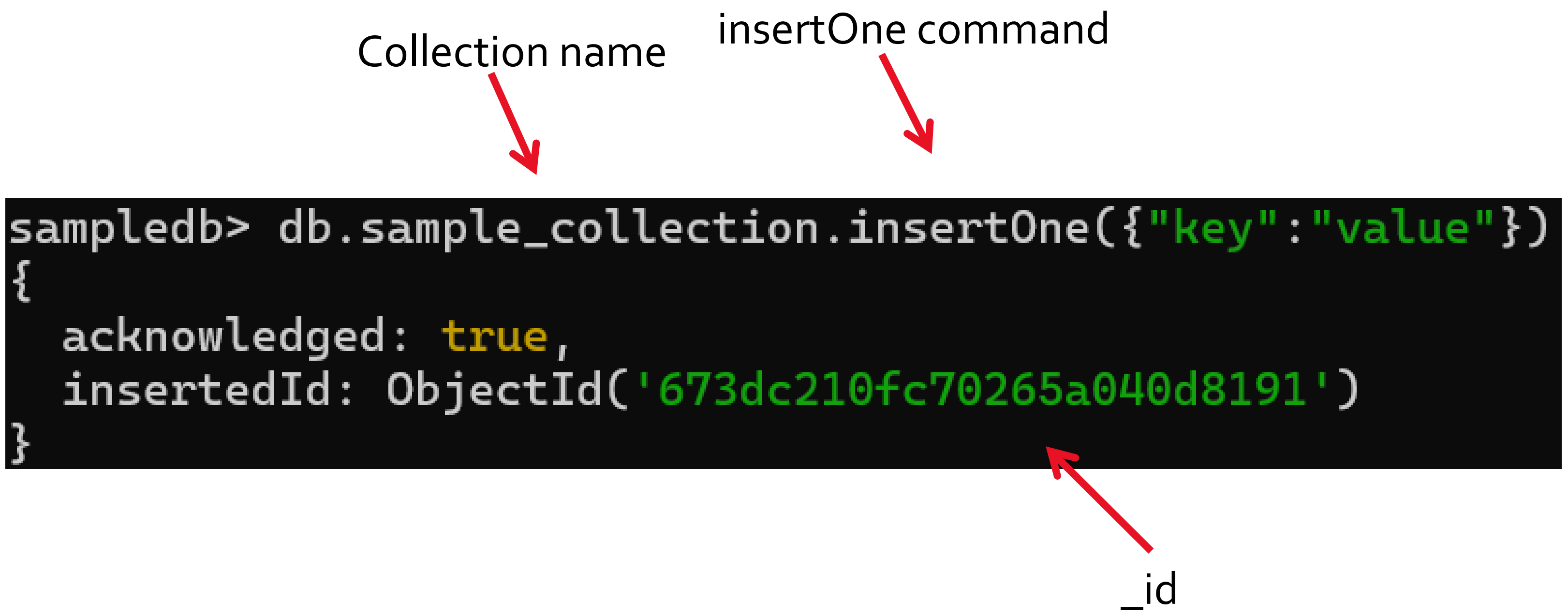
drop collection - to delete the collection from the databace



```
sampledb> db.sample_collection.drop()
true
```

Mongosh Commands- inserting documents

insertOne



```
sampledb> db.sample_collection.insertOne({"key":"value"})
{
  acknowledged: true,
  insertedId: ObjectId('673dc210fc70265a040d8191')
}
```

insertMany

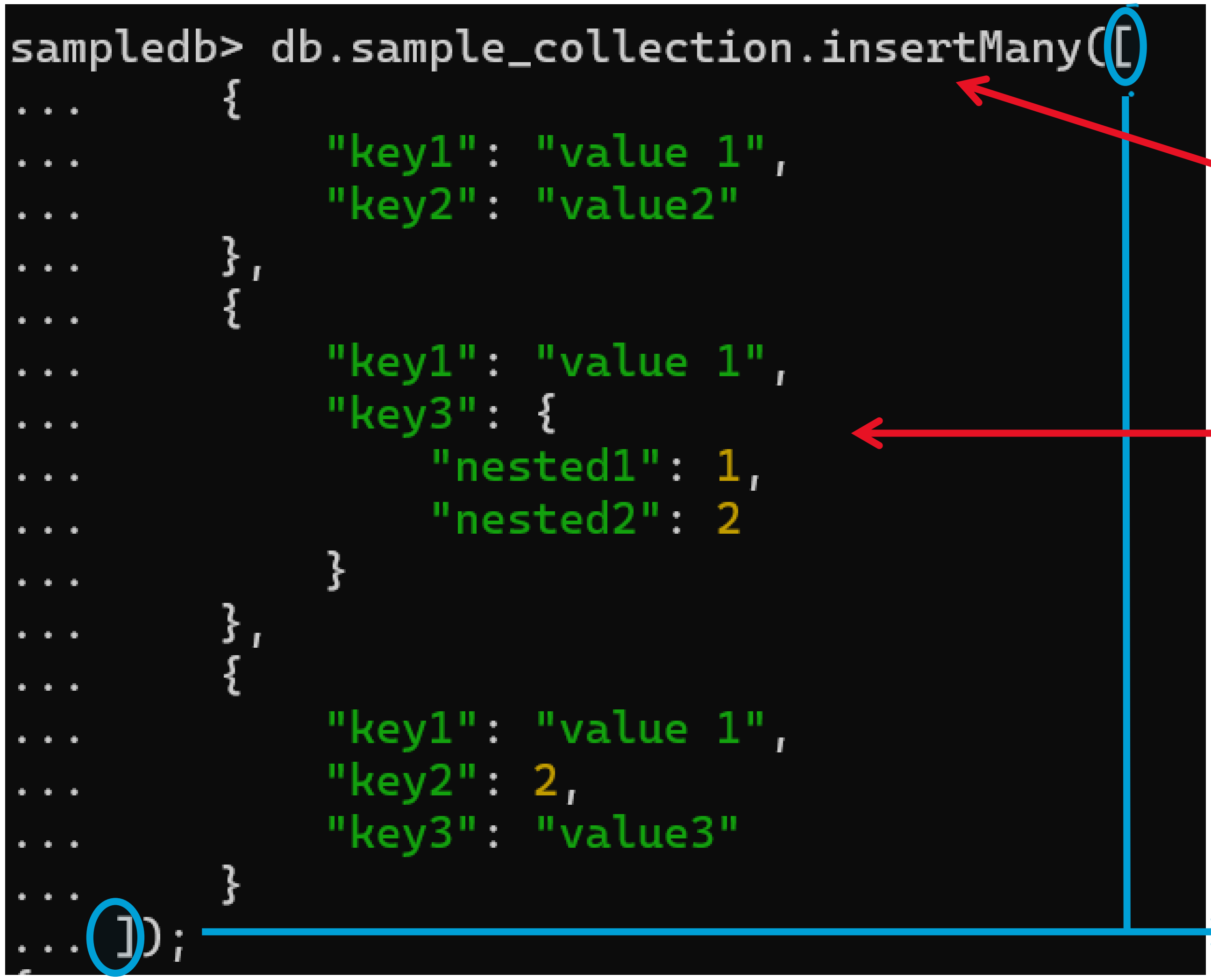
insert

Mongosh Commands- inserting documents

insertOne

insertMany

insert



```
sampledb> db.sample_collection.insertMany([
...       {
...           "key1": "value 1",
...           "key2": "value2"
...       },
...       {
...           "key1": "value 1",
...           "key3": {
...               "nested1": 1,
...               "nested2": 2
...           }
...       },
...       {
...           "key1": "value 1",
...           "key2": 2,
...           "key3": "value3"
...       }
... ]);
```

insertMany command

Different structurs are allowed

Array of documents

db.collection_name.insertMany([{},{},{}])

TECHNION
Azrieli Continuing Education and External Studies Division

Mongosh Commands- inserting documents

insertOne

insertMany

insert

```
db.collection_name.insert({info})
```

To insert only one document

```
db.collection_name.insert([{info},{info},{info}])
```

Use an array []
to insert many documents

Mongosh Commands- fetch

No filter

filter

findOne

db.collection_name.findOne()

db.collection_name.findOne({*key:value*})

find

db.collection_name.find() all

db.collection_name.find().limit(*num of results*) limit

db.collection_name.find ({*key:value*}) filter

db.collection_name.find ({},{key:true/false, key:0/1}) Choose fields

In this example the filter is empty, there is no filter

TECHNION
Azrieli Continuing Education and External Studies Division