**11-30 notes**

Introduction to NoSQL and MongoDB

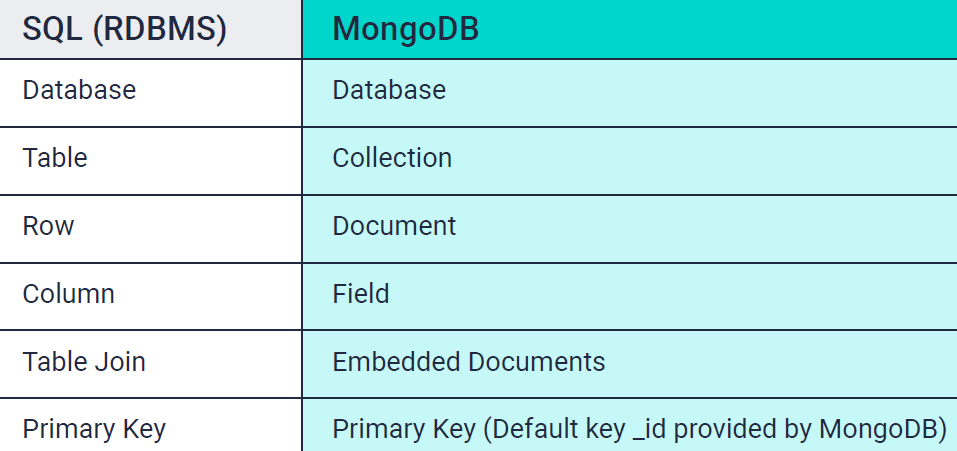
Goals:

* Identify the key differences between SQL and NoSQL databases to aid decisions around what kind of database to use in different situations.
* Create and connect to local Mongo databases.
* Perform create, read, update, delete (CRUD) operations on Mongo documents by using the Mongo shell.
* Import data from CSV and JSON files into a local MongoDB database.

NoSQL databases are effectively JSONs.

They excel at heterogeneous data formats and are easy to implement.

* You can pick just one (SQL or NoSQL) at the start of a project
  + or use one or the other for different tasks



^ You can also have sub-documents within a document.

**Advantages:**

● MongoDB uses a document data model

● MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL

● MongoDB stores data in RAM memory (binary format) which allows for quicker access.

● It has a flexible schema, or schema-free. Your code defines your schema.

● The syntax is easier to read and understand

● You can horizontally scale which helps increase storage capacity, whereas RDBM databases scale vertically and use a lot of memory.

**Disadvantages**

● You need a lot of RAM if you have large Mongo databases.

● It's difficult to create joins.

● Relationships are not well-defined

● Data can be duplicated

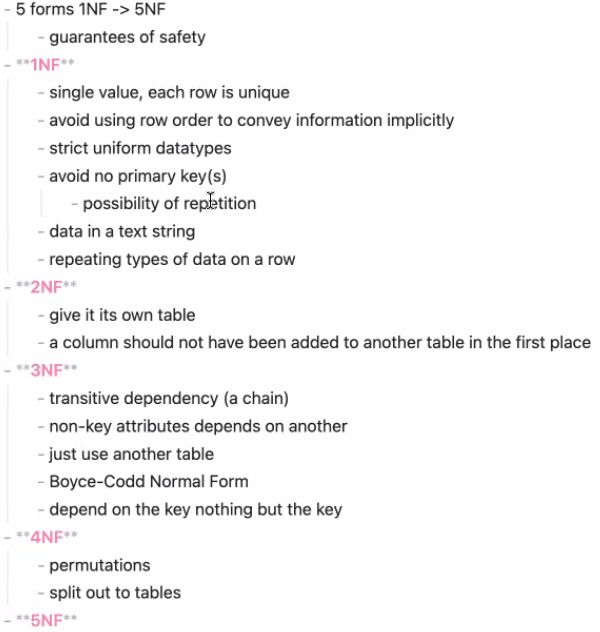
● Limited size is 16MB for a document

<https://www.mongodb.com/nosql-explained/nosql-vs-sql>

Lots of companies with customer-facing pages use MongoDB to store data.

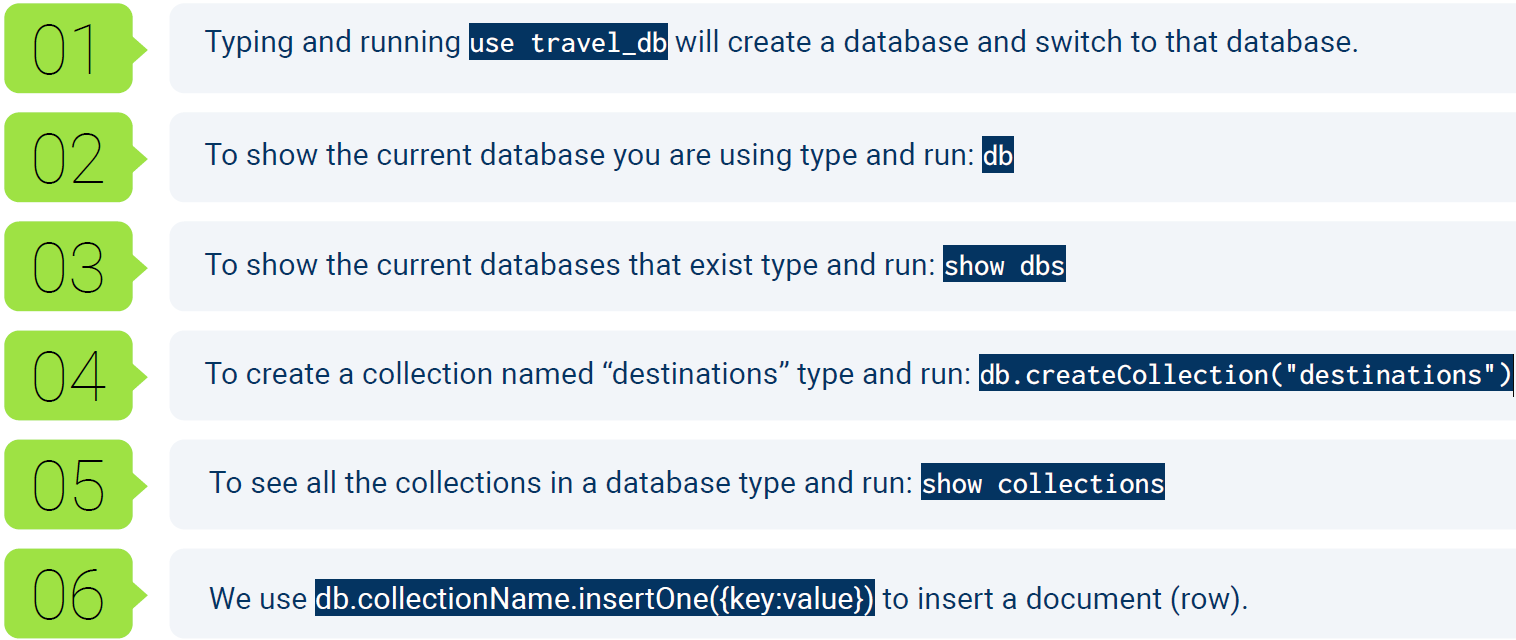
* It needs to be fast and have low ping, which MongoDB is great for.
* IE- Zoom (chat, reactions), Online gaming, Pinterest

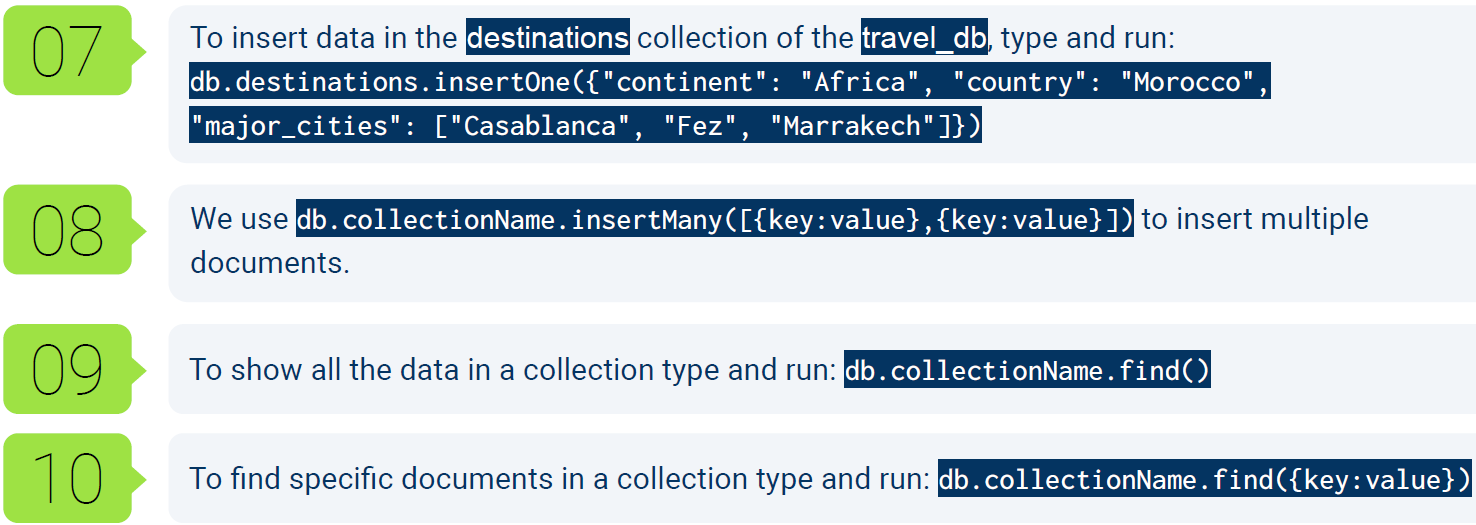
**Normal Forms:**



^ 3NF A=B, B=C, therefore A=C

**Using Mongo Shell (AKA Mongosh)**





**Commands within Mongo Shell:**

**Make a db:**

use DB\_name

**Make a table/collection within it:**

db.createCollection("collection\_name")

**View collections available to have data added, removed, or pulled from:**

show collections

**Adding data example:**

db.classroom.insertOne({'name': 'Khrystyne', 'favorite\_python\_library': 'unknown', 'age': '33', 'hobies': 'hunting', 'reading', 'knitting', 'video games'})

**Pull all records/documents from a collection:**

db.classroom.find()

**Pull one record/document from a collection:**

db.classroom.find({'name': 'Khrystyne'})

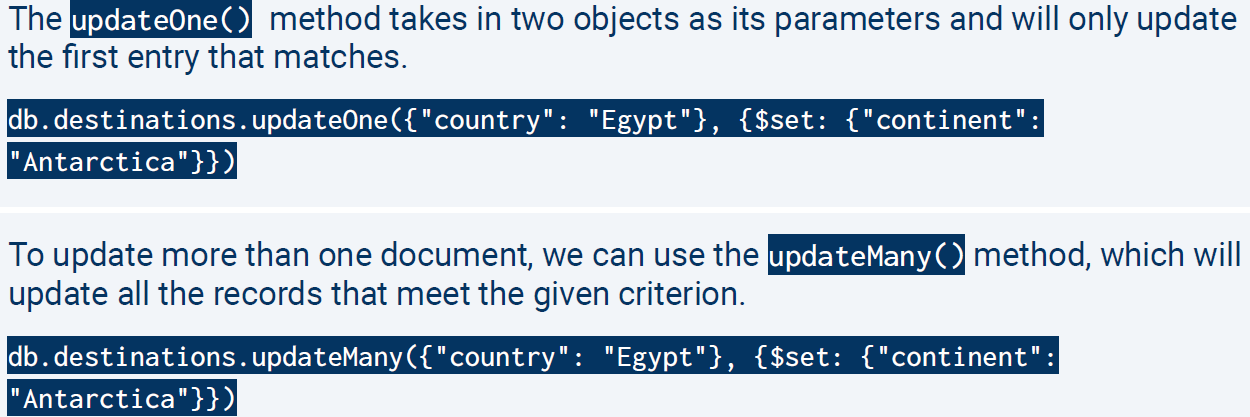
**Updating a record’s value in a field:**

db.destinations.updateOne({"country": "Egypt"}, {$set: {"continent": "Antarctica"}})

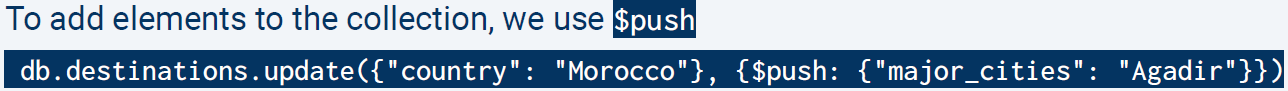
db.destinations.updateMany({"country": "Egypt"}, {$set: {"continent": "Antarctica"}})

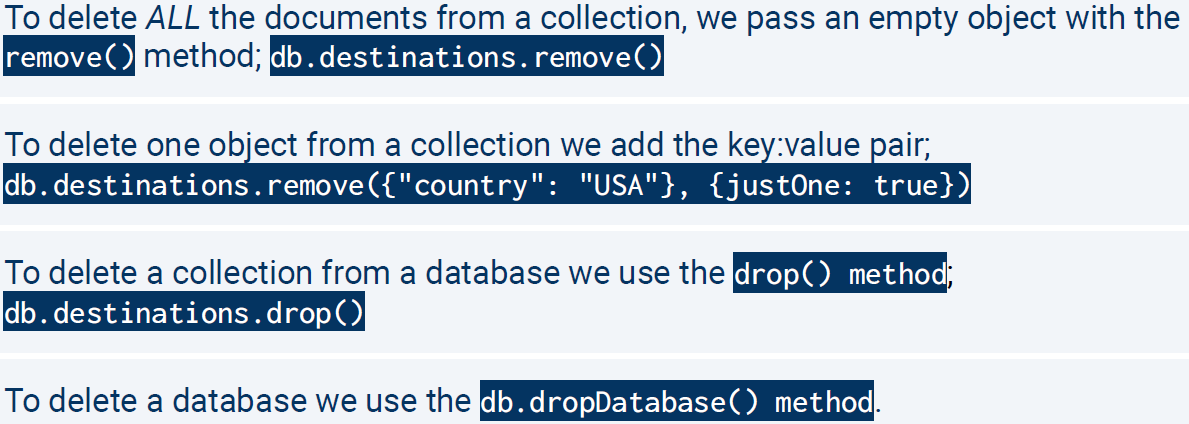
|  |  |
| --- | --- |
|  |  |

You can use Tab to auto-complete commands, like db.des TAB will do db.destinations



^ UpdateOne will update the first record, updateMany will update all





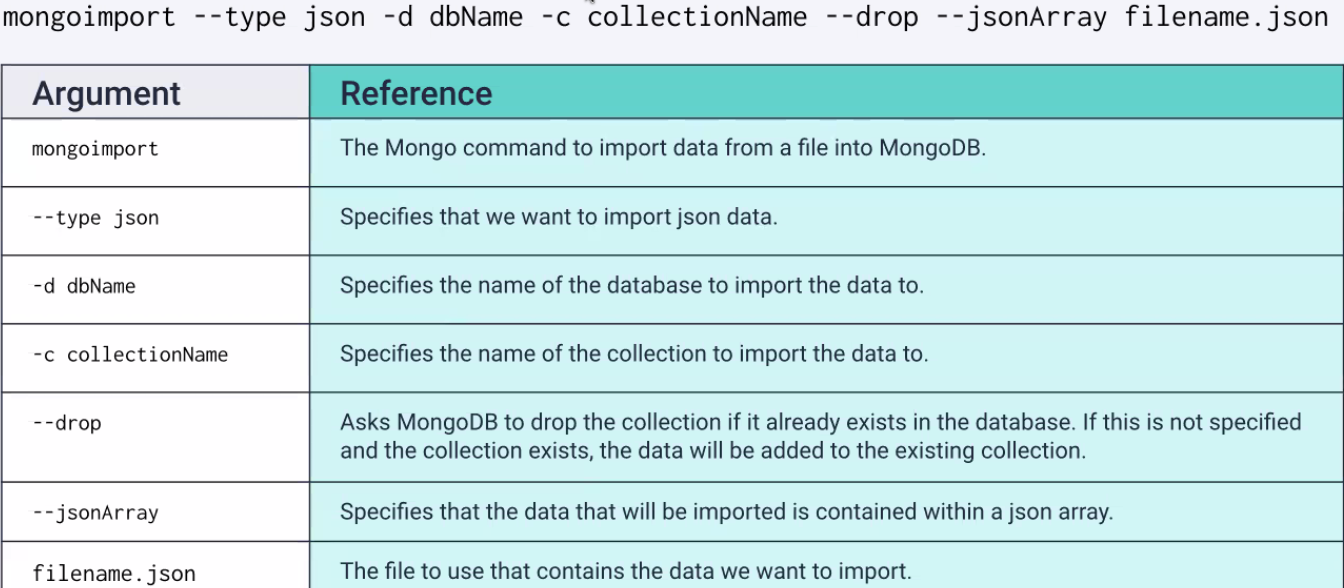
To add a record that doesn’t exist without using an insert, you can use an upsert.



Situations to use an upsert vs an insert- Like in a calendar app on your phone. Adding a new event

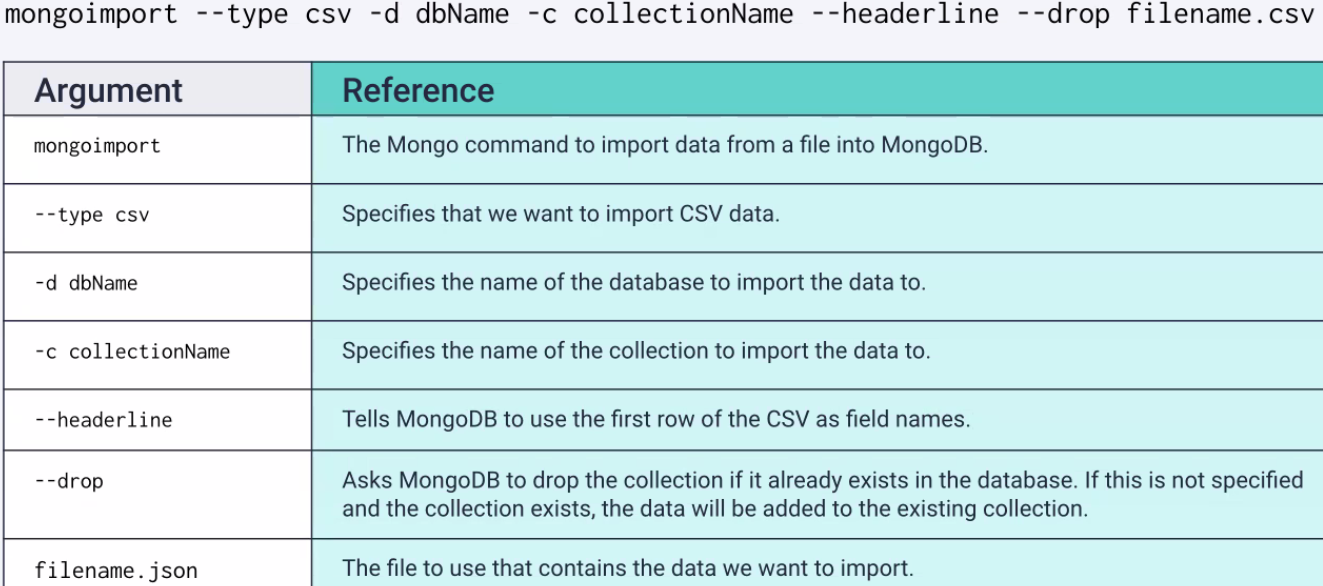
* If the event you’re trying to add doesn’t exist, an upsert will make the event.

**Importing Data from JSONs:**



Dashes with text to right are called Flags/options

**Importing Data from CSVs:**



**Importing command in powershell:** 