Connect to Mongo DB

Connection string

↙ Standard

↘ SRV format → Client → Mongosh

↓

DNS server ← DNS SRV record.

27017 → default port.

Basic components of connection string → Hostname of Mongo cluster & username & pass for authentication

to connect mongo to a diff port → mongosh --port 2888.

Postno.

① Network access errors → Check network access tabs to add IPaddresses.

M0, M2 & M5 clusters; maximum of 500 open connections

② Too many conn. error → (1) scale cluster to higher tier

(2) Restart Application

(3) Use maxPoolSize to limit the no of connection

③ User Auth error → Check if use exist & password & username correct.

Insert doc → insertOne()

Find value → .find()

Add & run JS code in mongosh

We can use external editor to edit fun with mongosh

Check if editor exists → config.get('editor')

Set editor → config.set('editor', 'vim')

Use → db.getSiblingDB() → to change databases within a script.

we can add validation in compass on required fields also there data types

ex. → { $jsonSchema: {
required: [
'name',
'stars'
],
properties: {
name: {
type: 'string'
},
stars: {
type: 'number'
}
}
}
}

on compass means even check if all data present passes these rules or not

in compass we can also use natural language query like

↓

find all documents that doesn't have "name" field

In compass we can check explain tab to see query performance before creating any index or something

$topN → We can use to get some top output of query like.

```
{ _id: "$cuisine",
  topRestaurants: {
    $topN: {
      n: 5,
      output: {
      name: "$name",
      stars: "$stars",
      restaurant_id: "$restaurants_id"
      },
      sortBy: {
      stars: -1
    }
  }
}
```

We can save aggregations for future use case

Connecting to MongoDB in Java.

Drivers are libraries required to interact with MongoDB.

Java Drivers

(1) Synchronous
(2) Asynchronous

Drivers → (1) Simplify connecting & interacting
(2) Establish secure connection to MongoDB cluster
(3) Execute DB operation on behalf of client applications
(4) Specify conn options → like security setting read/write ability

in maven for example.

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.7.1</version>
```

An application should use a single MongoClient instance for all database requests.

Instantiating a Client (MongoClient) instance is resource intensive

Use singleton design framework.

Always use MongoClient.close() → to close any open connections which may lead to too many open connections