**MongoDB in Local**

* Cross-platform document-oriented database
* JSON like syntax name BSON
* noSQL Database for apps development
* Fully manage in cloud
* 3x to 5x faster performance
* For large scalable apps
* Super security for better experience

**MongoDB install**

* To install mongoDB first download mongoDB community server with mongoDB Shell
* Now install mongoDB compass
* Now create a data folder in c drive with db
* Then add the bin path in system environment variables
* Now start mongoDB server with shell command   
  mongod - for server run  
  mongosh - for shell run   
  Exit - for turn off server

**SQL vs NoSQL**

* SQL Database system   
   -> Database   
   -> tables   
   -> col with rows ( data row )
* NoSQL Database System   
   -> Database   
   -> collections   
   -> bson type data like JSON ( document )

**Create Database & collection**

* Check database list   
  show dbs
* Check current database name   
  db
* create new database   
  use databaseName
* Delete Database   
  db.dropDatabase()
* Check collection list   
  show collections
* Create a collection   
  first use a database for collection then press command   
  db.createCollection(“collection name”)
* Delete any collection   
  db.collectionName.drop()
* Now **insert** a document ( data / multi-dymantional data ) to a collection   
  Insert - depricate now   
  insertOne  
  insertMany
* Get all document from a collection   
  db.collectionname.find(‘filter’, options) / .pretty()
* Get a **single** document  
  db.collectionName.findOne(‘filter’, options);
* **Update** document from a collection   
  updateOne(‘filter’, data, options)  
  updateMany(‘filter’, data, options)  
  replaceOne(‘filter’, data, options)
* **Delete** a document from a collection   
  deleteOne(‘filter’, options)  
  deletemany(‘filter’, options)
* **Find** the number of document into a collection   
  db.find().count() - deprecated   
  db.collection.countDocuments()
* **Some modifier of mongoDB**    
  Sort   
  Limit   
  Skip
* **MongoDB updates** operators   
  $set - update only det fields   
  $inc - increment the value of this field  
  $rename - rename field   
  $unset - remove field   
  $push - add a new value in a array   
  $pull - remove a value from an array  
  $pullAll - remove a value from an array  
    
  - **MongoDB search filter** operators   
  $eq - get all value with condition match  
  $ne - get all value with condition not match  
  $gt - get all value if the condition value is greater than   
  $gte - get all value if the condition value is greater than or equal   
  $lt - get all value if the condition value is greater than   
  $lte - get all value if the condition value is greater than or equal  
  $in - get all value within some common values  
  $nin - get all value without some common values  
  $and - get all data with name and email   
  $or - get all data with the value one or other one   
  $not - get all data without this value  
  $exists - get all data that have a field  
  $expr - get all values with the debet

**MongoDB Schema**

* **What is Schema ?**   
  A Schema is a model that represents your data structure in your database.
* There are three type id **schema** in mongoDB   
  -> Schema Less   
  -> Flexible Schema   
  -> Strict Schema
* **DataTypes in mongoDB**   
  -> String   
  -> Boolean   
  -> Number  
   -> NumberInt ( 32bit )  
   -> NumberLong ( 64bit )  
   -> NumberDecimal ( decimal )  
  -> ObjectId   
  -> ISODate  
  -> Embedded Document   
  -> Array

**Schema Validation**

* To manage strict Schema
* Maintain data easily
* Filter apps related data
* Create Schema Validator before create a collection
* Schema Flow   
  **Validator**  -> collection -> **validation** -> **Done** / **Fails**
* **Create a collection with validation**   
  db.createCollection(‘users’, {  
   validator : {  
   $jsonSchema : {  
   bsonType : “object”,  
   required : [“name”, ”age”, ”cell”],  
   properties : {  
   name : {  
   bsonType : “string”,  
   Description : “Name field is req”  
   },  
   age : {  
   bsonType : “int”,  
   Description : “Name field is req”  
   }  
   }   
   }  
   }  
  });
* **Schema Options**   
  -> bsonType   
  -> pattern   
  -> minLength  
  -> maxlength   
  -> required   
  -> description   
  -> min  
  -> max  
  -> [double]  
  -> enum
* Read more docs from [Check](https://www.mongodb.com/docs/manual/core/schema-validation/specify-json-schema/json-schema-tips/)

**Schema field keys**

* type
* required
* Default
* Unique
* Index
* Ref
* Select
* Lowercase
* Uppercase
* Trim
* minLength
* maxLength
* Populate
* Min
* Max
* Enum

**Queries**

* Sort
* Select
* And
* Or
* Equals
* Ne
* Gt
* Gte
* Lt
* Lte
* In
* Nin
* countDocuments
* estimateDocumentCount
* Exists
* Select
* Exec
* Limit
* Skip
* populate

**Custom Methods & Query**

* Mongoose custom method is used with a query
* It is too helpful create any kind of advance logic
* **To create a custom method**   
  Go to schema and make custom method   
  **userSchema**.**methods**.customMethodName = function(){  
   Custom query logics   
  }
* We can get all hooked data from **this** key
* **Static method**   
  **userSchema**.**statics.**.staticMethodName = function(p1, p2){  
   Custom query logics   
  }
* **Query methos**   
  **userSchema**.**query.**queryMethodName = function(p1, p2){  
   Custom query logics   
  }
* **Virtual property**   
  **userSchema**.**virtual**(‘propertyName’).**get**( function(){  
   Return output;  
  });

**Mongo middlewares**

* Mongo middlewares are some function which are run before or after a successful schema action
* There are two type of middleware   
  -> pre   
  -> post
* **Create a pre middleware**   
  **userSchema**.**pre**( ‘save’ , function(next){  
   next();  
  });
* **Create a post middleware**   
  **userSchema**.**post**(‘save’, function(doc, next){  
   next();  
  });

**mongoDB Cloud**

* Create an **Organization**-> Create a **project**   
   -> Create a **cluster**   
   -> Create a **database**   
   -> Create a **collection**  
   -> Create a **data**

FAOFYSYxxObPUCuy

wAXFNWbGh1OqhLcp

**mongoDB workFlow**

* Cluster
* Database
* Collection
* JSON formate



**Mongoose for mongoDB**

* a straight-forward, schema-based solution to model your application data
* elegant [mongodb](https://www.mongodb.com/) object modeling for [node.js](https://nodejs.org/en/)
* writing MongoDB validation, casting and business logic boilerplate
* It includes built-in type casting, validation, query building, business logic hooks and more, out of the box
* **Install mongoose**   
  npm install mongoose
* **Create a mongoDB Connection**   
  const mongoose = require(‘mongoose’);  
    
  const connectDB = async () => {  
   try {  
   const connect = await mongoose.connect(**MONGO**);  
   console.log(“MongoDB Connected : ${   
   connect.connection.host }”);  
   } catch(error) {  
   console.log(error);  
   }  
  }   
  module.exports = connectDB;
* **Initialization of Mongo Conneection**   
  Into server file just load this connection and invoke it   
  const connectDB = require(‘connectDB’);  
  connectDB();

**Connection String Customiztion**

* **To connect application**   
  mongodb+srv://haq47:<password>@facebook-database.zziwp.mongodb.net/<database\_name>?retryWrites=true&w=majority
* **To connect mongodb compass**

mongodb+srv://haq47:<password>@facebook-database.zzi  
 wp.mongodb.net/<database\_name>

**Data Model methods**

* **Search Data**find   
  findById  
  countDocuments  
  findOne  
  Exists  
  Where
* **Data Update**findOneAndUpdate   
  Update - deprecated   
  updateOne  
  updateMany
* **Create Data**   
  Create ( insert one & many )  
  Save   
  Insert - deprecated   
  insertOne - deprecated  
  insertMany
* **Delete Data**Delete - deprecated  
  deleteOne  
  deleteMany   
  Remove   
  replaceOne   
  findByIdAndDelete   
  findByIdAndRemove   
  findOneAndDelete  
  findOneAndRemove  
  findOneAndReplace

**Project**

* Create a complete student Data CRUD
* Create a complete Education Board Result System App
* Complete Meal system app