Mongo DB=>

 # 3 ways to work with mongo DB

1.Compass App - GUI tool.

2.VS code extension.

3.Through command shell

Step1 => Navigate to mongo DB bin folder path

C:\Program Files\MongoDB\Server\4.4\bin>.\mongo.exe

run the command .\mongo.exe

-> Commands through command shell.

***TO show all databases**

show databases

*** Show currently running database.**

db

***TO Create-**

use <database name>

***TO Drop Databases-**

db.dropDatabase()

***Creating and dropping collections**

db.createCollection(name, options)

db.collection.drop()

***Data types in mongo DB**

• Data Types in MongoDB

- BSON
- JSON
- Integer
- Boolean
- Double
- Arrays
- Object
- Null
- Date
- Timestamp
- Object Id
- Code

• Difference between BSON and JSON

ARC Tutorials

- JSON based databases usually return query results which can be effortlessly parsed, having modest or nix transformation, straightforwardly by the use of JavaScript along with most well-liked programming languages.
- In the case of MongoDB, data representation is done in JSON document format, but here the JSON is binary-encoded, which is termed as BSON.
- BSON is the extended version of the JSON model, which is providing additional data types, makes performance to be competent to encode and decode in diverse languages and ordered fields.

- **What is BSON?**
  MongoDB data type
  ⇨ Store and progress data
  ⇨ Binary encoded JSON – BSON => and it has some extended data types which are not supported by JSON. Below are the examples.
  => date
  => timestamp
  => object ID

- **Document Insertion in Mongo DB collection**

  - **db.collectionName.insert()**
  - **db.collectionName.insertMany()**

  -We can insert any number of documents into the collection.
  -Every document we insert will have a unique key "_id"
  - The Value for this key is always unique and 24 character
  - _id as a primary key in your collection

  -Can we change the value of _id?
  Ans -Yes, we can change it but it is not a good practice.

- **Update Existing Document in a Collections.**

  -update()
  -updateOne()
  -updateMany()

  Structure=>
  databasename.collectionName.update(
  {condition}, {value to set})

  Example:
  Database Name = Inventory
  Collection Name = users

```
Inventory.users.update(
{"name": "jaydeep"},
{
  $set: {
        "isActive": "true",
        "mobile" : "9067211658"
})
```

- **Reading Data from collection.**
  - **Multiple ways to Read data from collection**
    - **find**() – finds all documents in collection
      - db.collection.find()
    - **findOne**() – find first document in collection
    - **find**({"key1":"value", "key2": "value2"}) – by setting query conditions
    - **findOneAndReplace**({"key1":"value", "key2": "value2"}, <replacement>)
    - **findOneAndDelete**({"key1":"value", "key2": "value2"})

- **Delete Documents from Collection**

  - **Multiple ways to Delete data from collection**

    ARC Tutorials

    - **deleteOne**() – finds all documents in collection
      - db.collection.find()
      - Example: db.orders.deleteOne({ "_id" : ObjectId("563237a41a4d68582c2509da") });
    - **deleteMany**({})
      - Will delete many documents at once
      - When passed with empty curly brace – it will delete all documents in collections
      - Example: db.orders.deleteMany({});

- **Queries in MongoDB**
  - **Various conditions that can be used are:**
    - Equality
    - Less Than
    - Less than equal
    - Greater Than
    - Greater Than Equal
    - Not Equal

- $and - And operation
  - Match all conditions mentioned in the Find method
  - E.g db.leads.find({ $and: [{}, {}] })

- $or - OR operation
  - Match any condition in the find method

```
1. find() - will return all the documents in the collection
2. Find with conditions
    db.leads.find({"Tax": "30"})
3. Find method with multiple conditions ( AND) by default
    db.leads.find({"Tax": "30", "Salary": "120000"})

4. Find method with tax less than 30
    db.leads.find({"Tax": { $lte: "30" }})

5. Find method with tax grater than equal 30
    db.leads.find({"Tax": { $gte: "30" }})
```

```
6. Find method with $eq
    db.leads.find({"Tax": "30"})

7. And operator with Find method
    db.leads.find({$and : [{"Tax": "30" }, {"Salary": {$lte: "120000"}} ]});

8. OR operator with Find method
    db.leads.find({$or : [{"Tax": "30" }, {"Salary": {$lte: "100000"}} ]});

9. And and OR inside a Find method

10. And, OR, lte, gte, eq inside Find method
```

- **Find Specific Fields in MongoDB**

- **We need to specify the field as "1" or "0"**
  - db.leads.find({<condition>},{"Tax":1, "_id": 0})

```
Selecting Fields
    db.leads.find() - all documents are returned

    db.leads.find({},{"city":1}) - get city key from all documents in the collect
        - it most cases you will need this _id for processing needs
        - click/delete/remove/edit -> unique value that value is _id

    db.leads.find({}, {"city": 1, "_id": 0 })

    db.leads.find({"Tax": {$lte: "30" }}, {"Tax":1,"city":1, "_id": 0})
```

- **Projection In mongo DB**

  -By Default it will bring up all keys/value s from all documents in collections
  -Drill it down
  db.leads.find({“Tax”:30)}

  -number of documents will reduce
  -60 keys
  -We do not need all keys
  We need only few keys

  For Example => 1. SELECT * FROM “table” SQL/RDBMS
  2. Find() => All documents.

```
find({}, {"Tax":1, "_id":0, "leadName":1})
    -> MongoDB -> get all documents
        -> matching documents
            -> extract the keys that we have asked for
                -> this result set is returned/projected as output
```

- **Aggregation**

  - **What is Aggregation in MongoDB?**

    - Aggregate is very similar to the find command, where you can provide the criteria for your query in the form of JSON documents

    - The key element in aggregation is called the pipeline

    - It also helps us in performing few operations like min, max, sum etc

    - The command to use **Aggregation** is :
      - **db.leads.aggregate(pipeline, options)**

      - **What's pipeline?**
        - A sequence of data aggregation operations or stages
        - Pipeline is an Array

      - **What are options?**
        - Documents can be passed as well

- **What are valid Aggregate Stages?**
  - $count
  - $group
  - $limit
  - $lookup
  - $match
  - $merge
  - $sort
  - $project
  - $unwind
  - $unset
  - And many more

- **Pipeline definition**

```
pipeline = [
    { . . . },
     { . . . },
     { . . . },
];
```

Example:

```
var pipeline = [
      { $group: { "_id": "$city" } }
      { $sort:    { "leadName": 1   } },
      { $limit: 4 }
    ];

db.leads.aggregate(pipeline);
```

- **Limit and Skip**
  - We may not always want all documents all the time
    - **db.collection.find().limit(4);**
  - We may want to skip some documents we don't need
    - **db.collection.find().skip(3);**
    - Always remember – skip will skip sequentially not random or advaoc

- **Sorting In MongoDB**

  - We will need to sort the record set before passing it to next logical operation
    - To sort we can use the below command
      - db.collection.find().sort({"leadName": 1 })
        - 1 : means ascending
        - -1 : means descending

- **Creating Indexes**

  - Indexes are the fastest way to find information
    - Relate it to Index page in your book?
  - Indexes concept is same as that you already would know in SQL
  - By default – every collection will have an Index on "_id" key
  - **How do we create a index on collection?**
    - db.leads.ensureIndex("leadName": 1)

- **Back Up and Restore**

  - **Steps to create Back up of MongoDB Data**
    - **Step #1** – First create a folder where you want to save your data
    - **Step #2** – Next, Runt the command "mongodump.exe"
    - **Step #3** – Verify the data dump and folder dumped correctly