MONGO DB

[{

        "name": "Sita",

        "age": 25,

        "Hobbies": [

          "Walk",

          "Cricket"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I am a youtuber",

        "experience": [

          {

            "company": "Spotify",

            "duration": 3

          },

          {

            "company": "Paytm",

            "duration": 1

          }

        ]

      },

      {

        "name": "Shyam",

        "age": 11,

        "Hobbies": [

          "Walking",

          "Reading"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I am a youtuber and actor"

      },

      {

        "name": "Ghanshayam",

        "age": 11,

        "Hobbies": [

          "Walk",

          "Cricket"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I am a cook",

        "experience": [

          {

            "company": "Times Internet",

            "duration": 1

          },

          {

            "company": "1 mg",

            "duration": 1

          },

          {

            "company": "Apple",

            "duration": 2

          }

        ]

      },

      {

        "name": "Rita",

        "age": 11,

        "Hobbies": [

          "Anime"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I play games"

      },

      {

        "name": "Ram",

        "age": 10,

        "Hobbies": [

          "Walk",

          "Cricket"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I do nothing.",

        "experience": [

          {

            "company": "KPMG",

            "duration": 1

          },

          {

            "company": "EY",

            "duration": 1.5

          },

          {

            "company": "TCS",

            "duration": 0.5

          }

        ]

      },

      {

        "name": "Geeta",

        "age": 12,

        "Hobbies": [

          "Gaming",

          "Cooking"

        ],

        "identity": {

          "hasPanCard": false,

          "hasAdhaarCard": true

        },

        "bio": "I code and play games.",

        "experience": [

          {

            "company": "Spotify",

            "duration": 1

          },

          {

            "company": "Google",

            "duration": 3.4

          }

        ]

      },

      {

        "name": "Geeta",

        "bio": "I just code.",

        "experience": [

          {

            "company": "Amazon",

            "duration": 2

          },

          {

            "company": "Flipkart",

            "duration": 1

          }

        ],

        "Hobbies": [

          "Reading",

          "Walking"

        ]

      },

      {

        "name": "Akshit",

        "age": 22,

        "hobbies": [

          "TV Shows"

        ],

        "hasMacBook": true,

        "bio": "I am savage boi.",

        "experience": [

          {

            "company": "Amazon",

            "duration": 2

          },

          {

            "company": "Google",

            "duration": 3

          }

        ]

      }]

 const data = await model1.find({'identity.hasAdhaarCard':true})

        res.status(200).json({data:data})

    } catch (error) {

        res.status(400).json({error:error})

    }

 const data = await model1.find({age:{$nin:[25,22]}})

// const data = await model1.find({age:{$in:[25,22]}})

// const data = await model1.find({age:{$gt:[25,22]}})



* $eq : equal to
* $ne :not equal
* $lt : less then:
* $gt : greatet then
* $lte:less then equal to
* $gte : greater then equal to
* $in : check it present in list like we find age [22,25]
* $nin : not in list

**Logical Operators in MongoDB in Hindi ( $not, $and, $or & $nor)**

const logicalOperator = async(req,res)=>{

  try {

    //find age less then 10 and greater then 12

    // const data =await model1.find({$or :[{age:{$lte:10}} ,{age:{$gte:12}}]})

    //nor operator

    // const data =await model1.find({$nor :[{age:{$lte:10}} ,{age:{$gte:12}}]})

    //using end operator   age less then 11 and hobbies = 'Walk'

    // const data =await model1.find({$and :[{age:{$lt:11}} ,{Hobbies:'Walk'}]})

    //direct 2 queries on age but it wrong  use $and in this json pick last quert age:{$gte:20}

    const data =await model1.find({age:{$lt:11} ,age:{$gte:20}})

      res.status(200).json({data:data})

  } catch (error) {

      res.status(400).json({error:error})

  }

}

# Mastering MongoDB: Understanding the $exists and $type Operators

// Mastering MongoDB: Understanding the $exists and $type Operators

const existOperators = async(req,res)=>{

  try {

    //check field hasMacBook present in any document

    // const data =await model1.find({hasMacBook:{$exists:true}})

    // const data =await model1.find({hasMacBook:{$exists:true,$eq:true}})

    //wo doc gen ma hasMacBook ki type Boolean ho

    const data =await model1.find({hasMacBook:{$exists:true,$type:"bool"}})

      res.status(200).json({data:data})

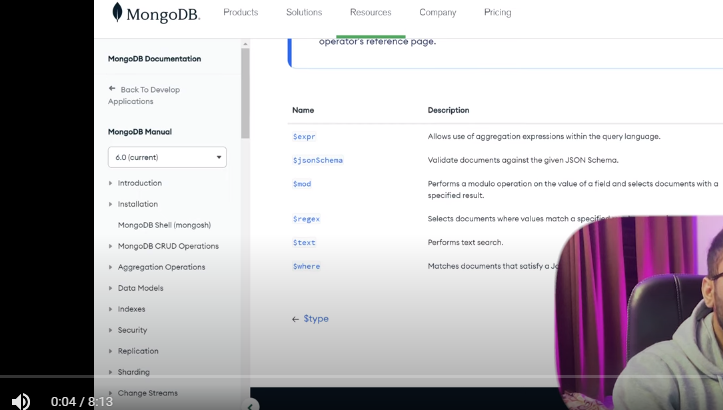
  } catch (error) {

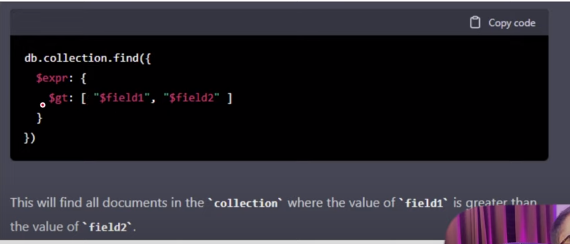
      res.status(400).json({error:error})

  }

}

# Evaluation Operators

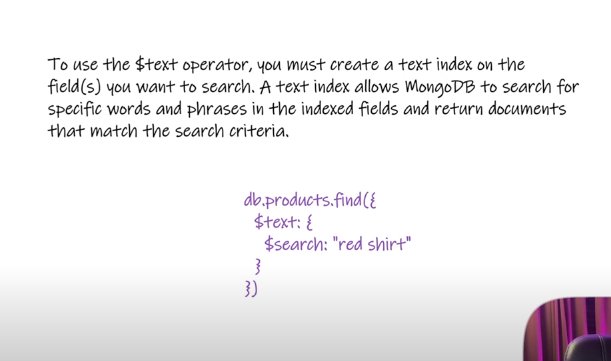
****

****



$regr : regular expression

$text: it’s a text search first index the filed you want to apply $text



# From Beginner to Pro: Querying Arrays in MongoDB

const arrayQuery = async(req,res)=>{

    try {

        // const data =await model1.find()

        //find who work in amazon

    //   const data =await model1.find({"experience.company":"Amazon"})

      //how many student who work in 3 compinies

    //   const data =await model1.find({experience:{$size:3}})

      // find student whose  expereince have 3  or more dcoc

//    const data =await model1.find({$and :[{experience:{$exists:true}},{$expr:{$gte:[{$size:"$experience"},3]}}]})

     //find student whose hobbies "Walking" or "reading   "

    //  const data =await model1.find({Hobbies:{$all:['Walking','Reading']}})

//find hobbies in this

     const data =await model1.find({Hobbies:{$in:['Walking','Reading']}})

     //document that have at least a product with name apple  and quantity 15

      return res.status(200).json({data:data})

    } catch (error) {

       return  res.status(400).json({ success:false,error:error})

    }

}

Working with product

const addProduct=async(req,res)=>{

    var list =[

            // "\_id":1,

       [

                {

                    "name":'apple',

                    "quantity":10

                },

                {

                    "name":'orange',

                    "quantity":5

                },

                {

                    "name":'banana',

                    "quantity":20

                },

            ],

            // "\_id":2,

            [

                {

                    "name":'apple',

                    "quantity":15

                },

                {

                    "name":'orange',

                    "quantity":10

                },

                {

                    "name":'banana',

                    "quantity":5

                },

            ],

    ]

    var lst= [

        {

            "name":'apple',

            "quantity":15

        },

        {

            "name":'orange',

            "quantity":10

        },

        {

            "name":'banana',

            "quantity":5

        },

    ];

    var doc ={product:lst}

    try {

        const d= await productModel(

{

    product:lst

}

        )

var data= await d.save()

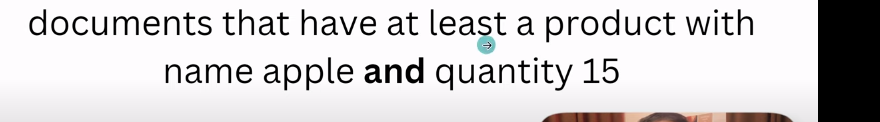
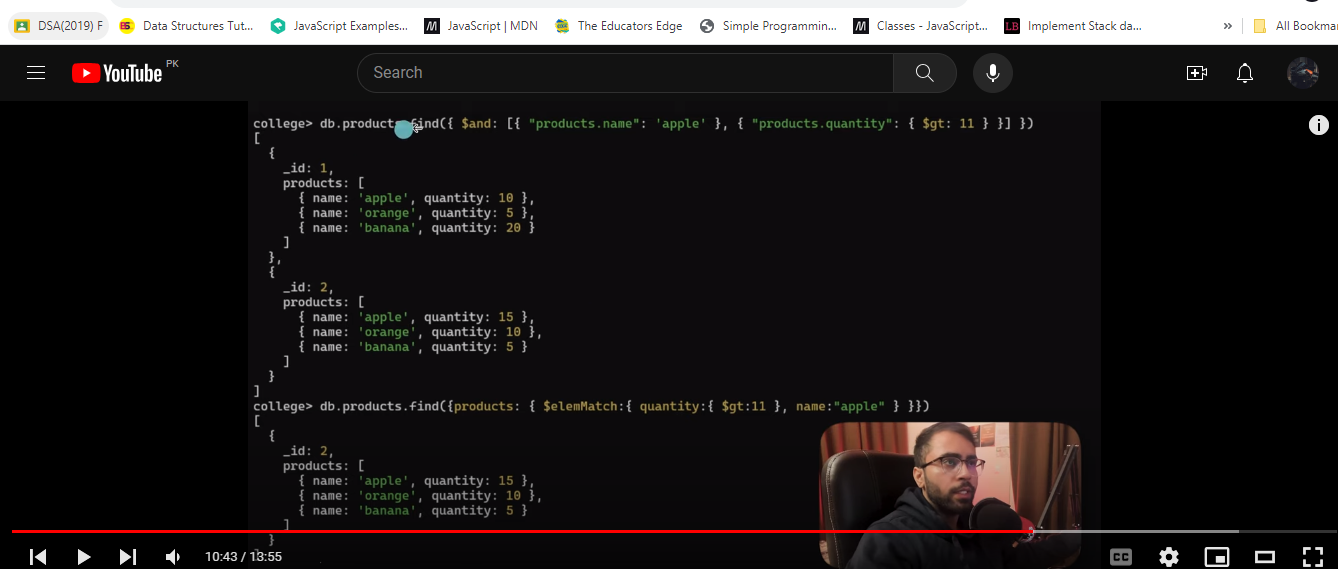
        return    res.status(200).json({data:data})

    } catch (error) {

        return  res.status(400).json({ success:false,error:error})

    }

}



SORT

const addTeacher = async (req, res) => {

    var doc = { name: "jarry", age: 32, gender: 'male' }

    try {

        const d = await teacherModel(doc)

        var data = await d.save()

        return res.status(200).json({ data: data })

    } catch (error) {

        return res.status(400).json({ success: false, error: error })

    }

}

const QueryOnTeacher = async (req, res) => {

    try {

        //assending =1  small to large

       // desending=-1   lage to small

       //when age is same sort with name

       //inital mongo print only 20 result

        // const data = await teacherModel.find().sort({age:1,name:1})

        // const data = await teacherModel.find().sort({age:1,name:1}).forEach(x => printjson(x))

        // const data = await teacherModel.find().sort({age:1,name:1}).limit(5)

        const data = await teacherModel.find().sort({age:1,name:1}).count()

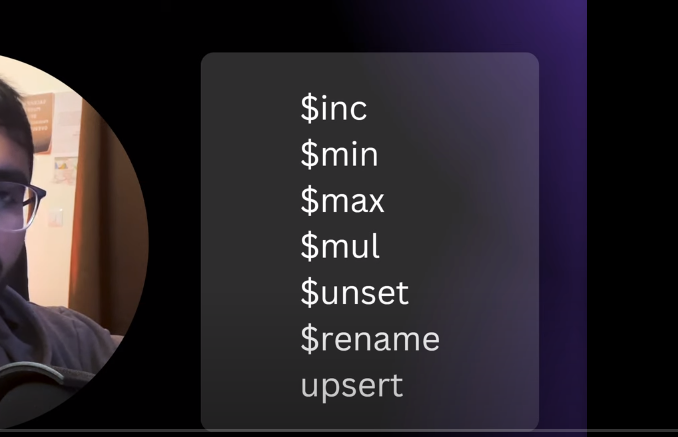
        return res.status(200).json({ data: data })

    } catch (error) {

        return res.status(400).json({ success: false, error: error })

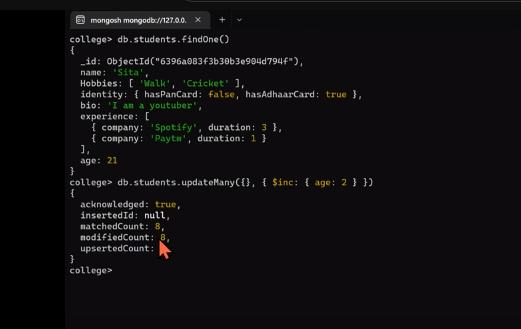
    }

}

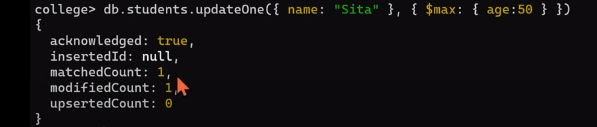


$inc : Increment

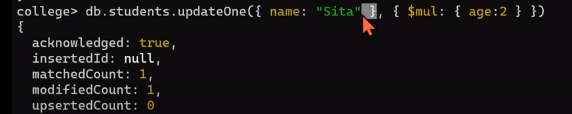
Update all student age by 2







**$mul means Multiply**

****

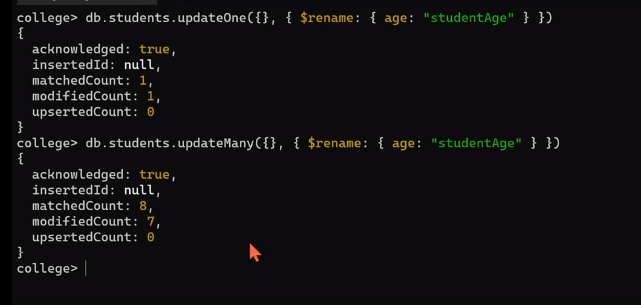
**$unset means delete field**

****

**$set add field**

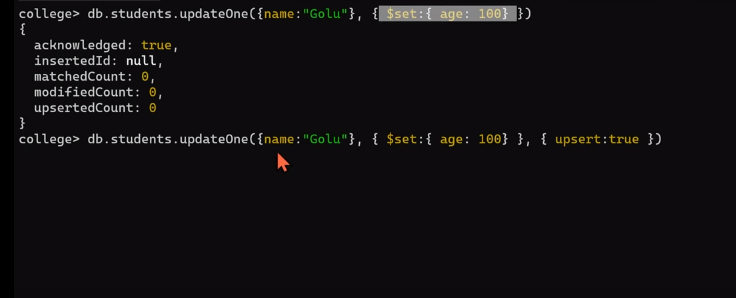
****

**$rename rename field**

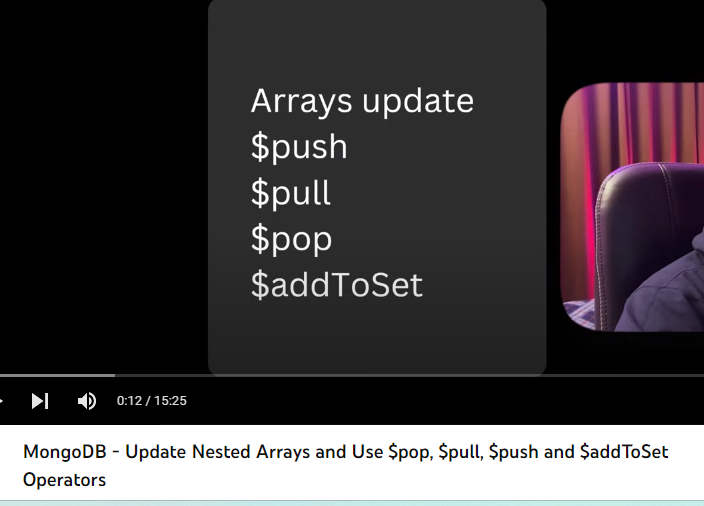
****

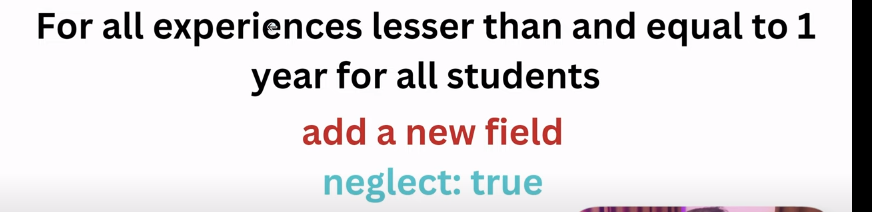
**$upset :measn find and set new age value**

**If Golu not find insert new**

****

**NESTED ARRAY vid 27**

****

****

const nestedArrayOperation = async(req,res)=>{

    try {

    //update only first field and  add neglect:true

    //    const data =await  model1.updateMany({experience:{$elemMatch:{duration:{$lte:1}}}  } ,

    //     {$set:{"experience.$.neglect":true}}

    // )

        //update all  field and  add neglect:true

    //    const data =await  model1.updateMany({experience:{$elemMatch:{duration:{$lte:1}}}  } ,

    //     {$set:{"experience.$[].neglect":1}}

    // )

           //update only mateched  field and  add neglect:true

       const data =await  model1.updateMany({experience:{$elemMatch:{duration:{$lte:1}}}  } ,

        {$set:{"experience.$[e].neglect":true}},

        {arrayFilters:[{"e.duration":{$lte:1}}]}

    )

//  const data=await model1.find({experience:{$elemMatch:{duration:{$lte:1}}}  })

        res.status(200).json({data:data})

    } catch (error) {

        res.status(400).json({error:error})

    }

}

Add object in experience

$addToSetset

$push

$pull

const addMoreObjectinExperience = async(req,res)=>{

    try {

//add Object in array   using $push

    //    const data =await  model1.updateOne({name:"Ram"},{$push:{experience:{company:"Meta",duration:2}}})

    // /if not present add  Object in array   using $addToSetset

    //    const data =await  model1.updateOne({name:"Ram"},{$addToSet:{experience:{company:"Meta",duration:2}}})

        // rempove  Object in array   using $addToSetset

        // const data =await  model1.updateOne({name:"Ram"},{$pull:{experience:{company:"Meta",duration:2}}})

          // rempove  last Object in array   using $pop

            //   const data =await  model1.updateOne({name:"Ram"},{$pop:{experience:1}})

                      // rempove  first Object in array   using $pop

            //   const data =await  model1.updateOne({name:"Ram"},{$pop:{experience:-1}})

         const data=await model1.find({name:"Ram"})

        res.status(200).json({data:data})

    } catch (error) {

        res.status(400).json({error:error})

    }

}