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) {

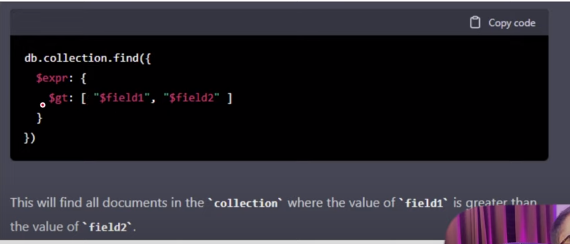
      res.status(400).json({error:error})

  }

}

# Evaluation Operators

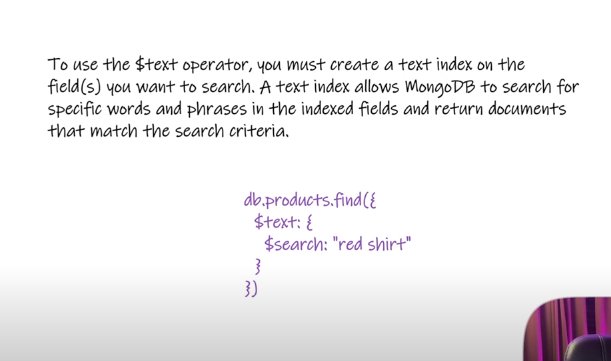
****

****



$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()

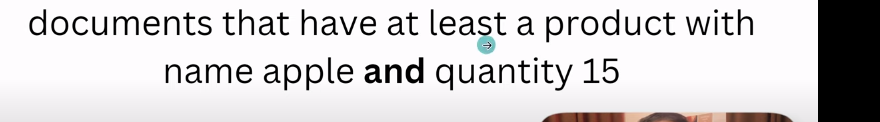
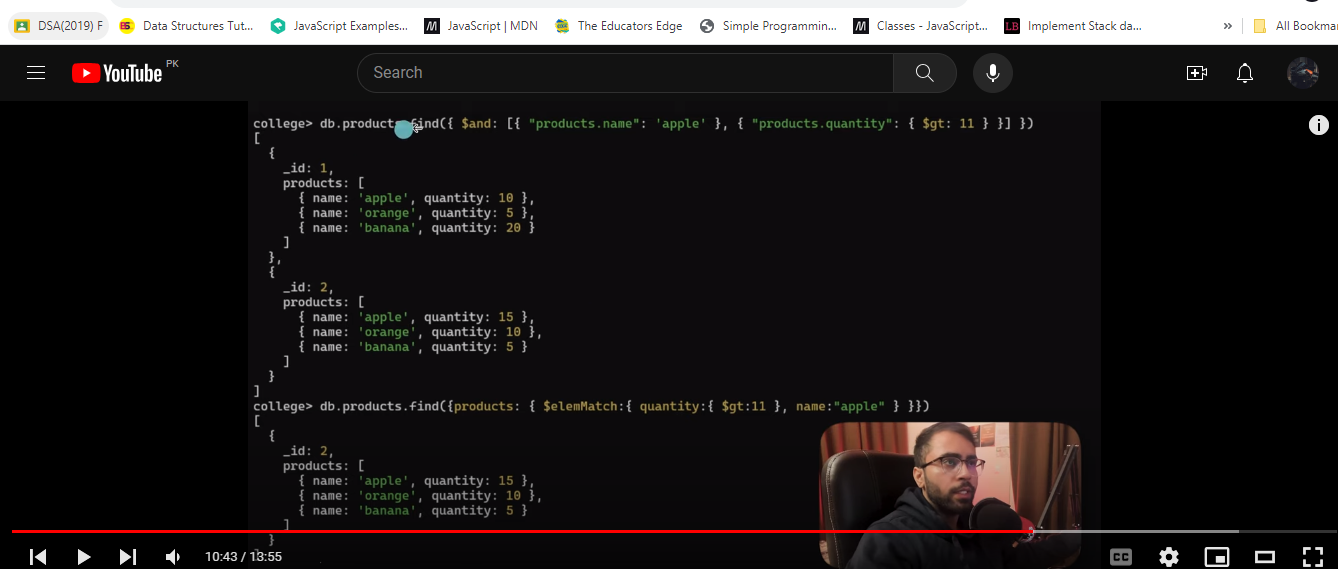
        return    res.status(200).json({data:data})

    } 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()

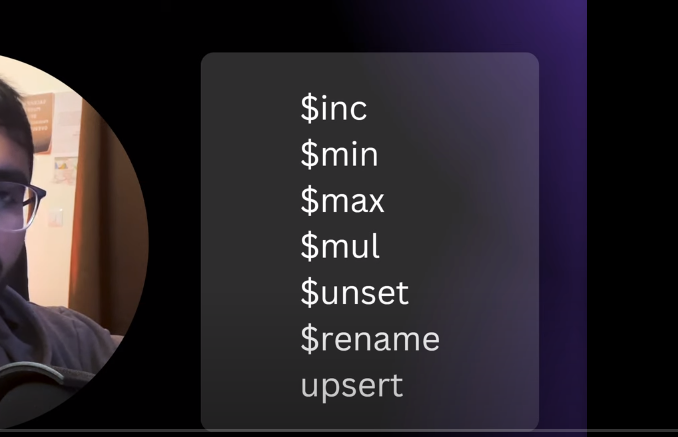
        return res.status(200).json({ data: data })

    } catch (error) {

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

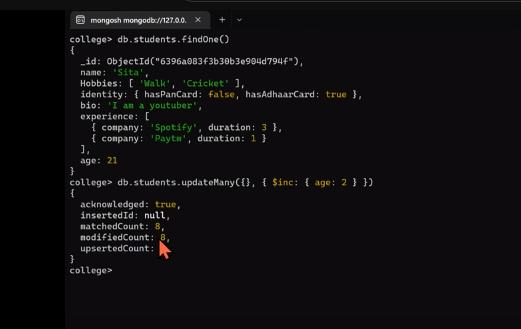
    }

}

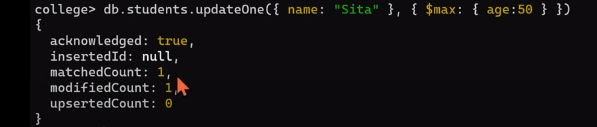


$inc : Increment

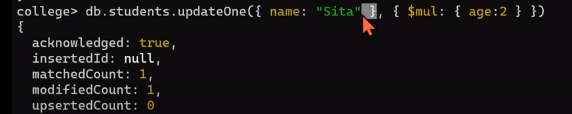
Update all student age by 2







**$mul means Multiply**

****

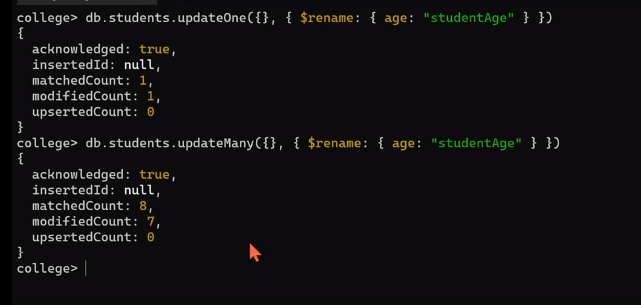
**$unset means delete field**

****

**$set add field**

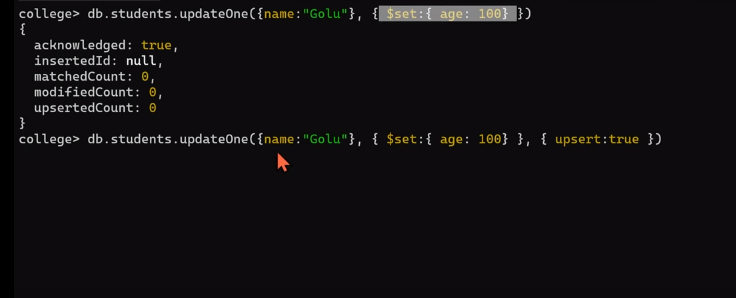
****

**$rename rename field**

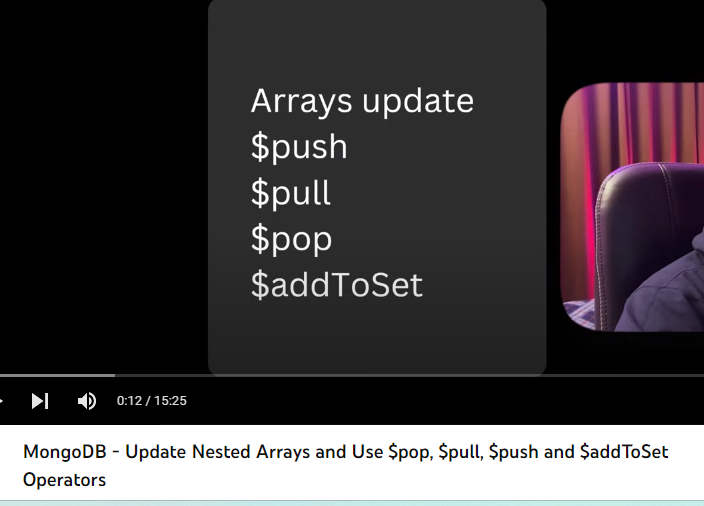
****

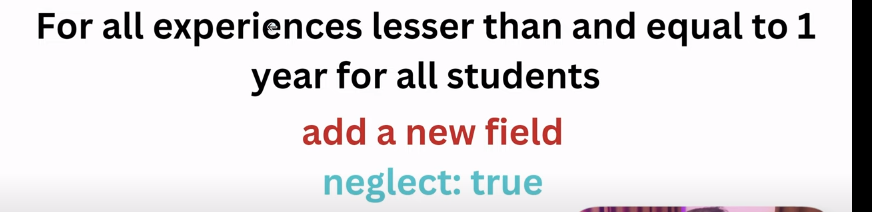
**$upset :measn find and set new age value**

**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})

    }

}

**Create Index**

**Create index with -1 descending 1 for assending**

**db.teachers.createIndex({"age":1})**

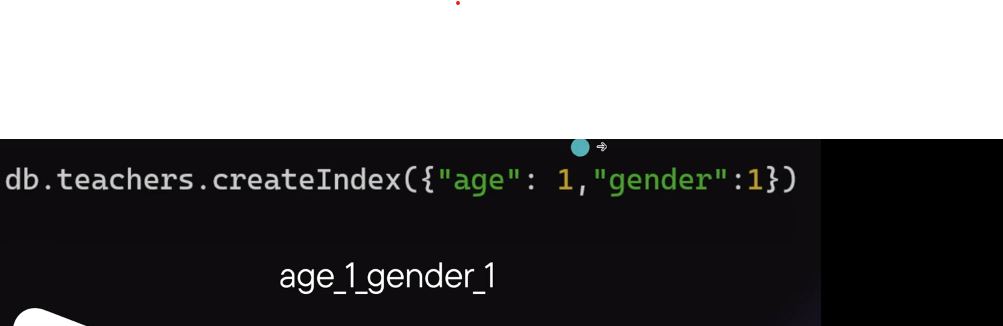
**db.teachers.getIndexes()**

**db.teachers.dropIndex({"age":1})**

**Compund index**

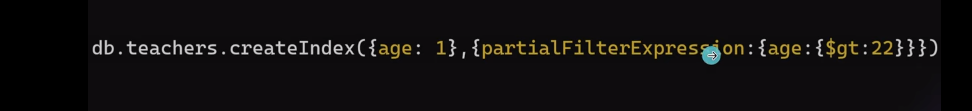
Is ma in dono ka combination sa index hota ha

In blow image order also matter



Partial Index

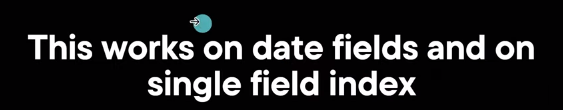
To index on perticuar doc



In this index only age greater then 22

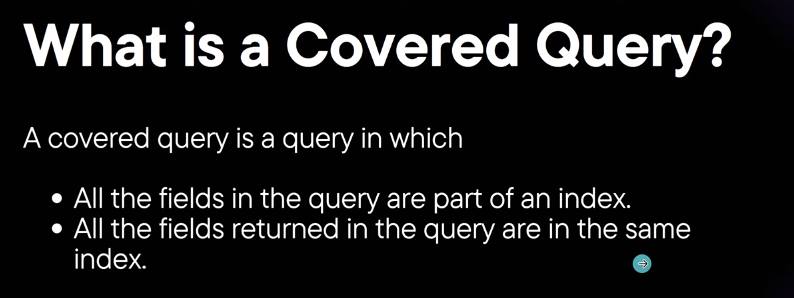


Use expire on date field index





**Covered Query**

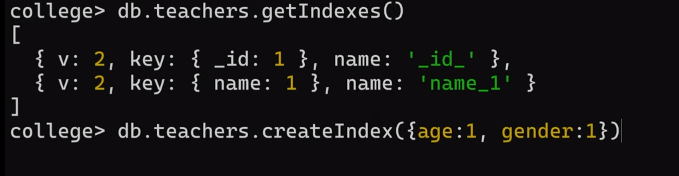




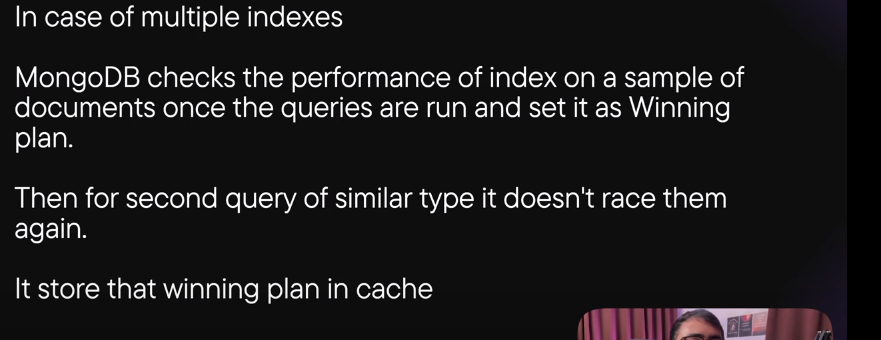
It return only name

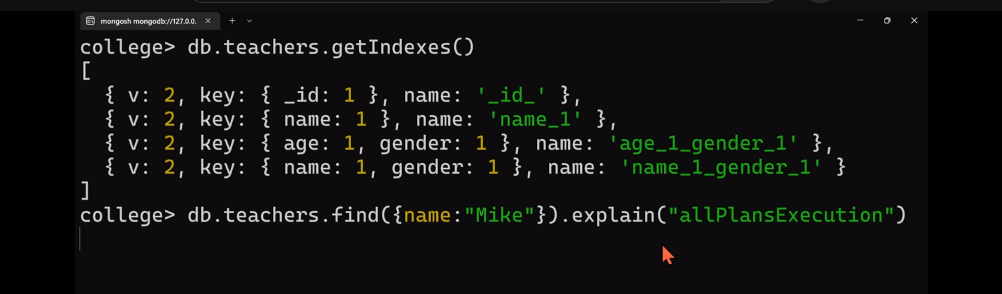
**Multiple Index**

Create index



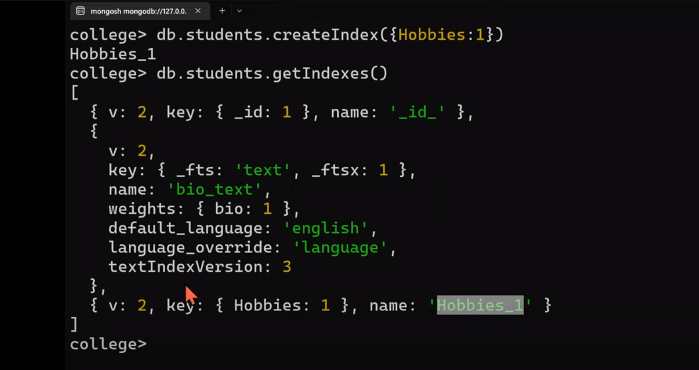






**MULTI KEY INDEX**

create index on array



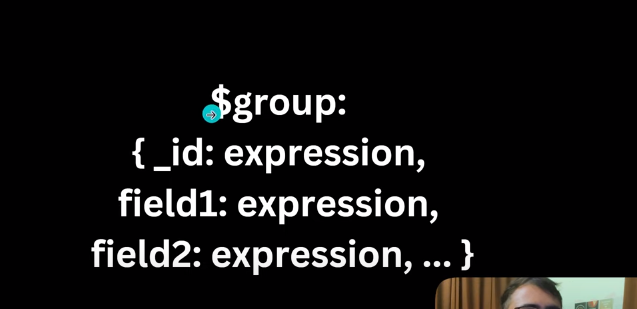
**23 min vid complete  totoal 38 min**

**AGGRIGATE Vid 29**

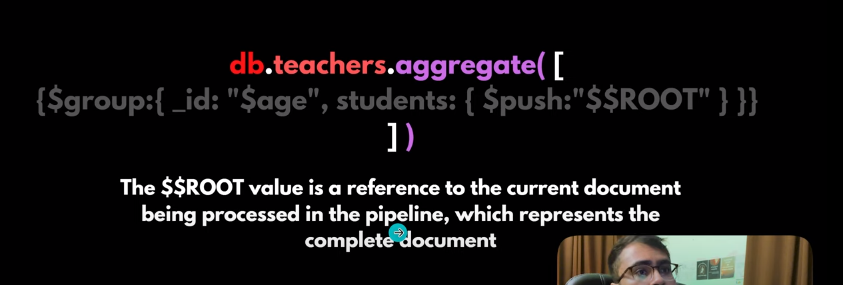
****

****

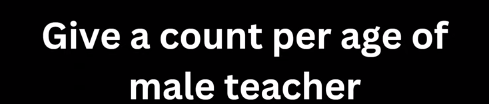
$group sintex



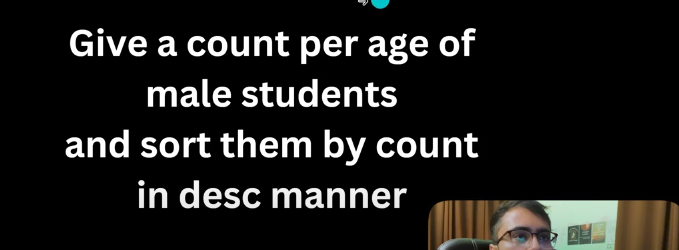
Group doc with age and send complete user doc



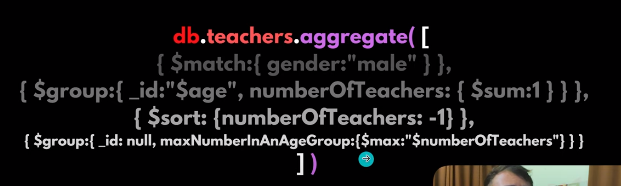
const data =await teacherModel.aggregate([{$group:{\_id:'$age',puraDocument:{$push:"$$ROOT"}}}])



    const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}}])

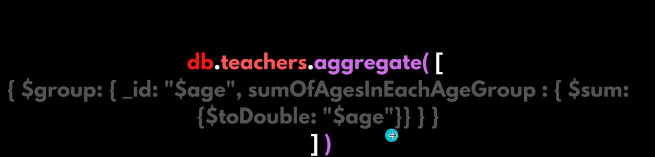


  const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}},{$sort:{countOfTeacherIngroup:-1}}])



    const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}},{$sort:{countOfTeacherIngroup:-1}},{$group:{\_id:null,maxNumberinAgeGroup:{$max:'$countOfTeacherIngroup'}}}])

**$toDouble**



 const data =await model1.aggregate([{$group:{\_id:"$age",hobbies:{$push:"$Hobbies"}}},])

$unwind

{

Hobbies: [‘hha’,’aaa’.’afafaf’]

\_id:1

}

Itconvert like this

{

Hobbies: ‘hha’,

\_id:1

},

{

Hobbies: ‘’aaa’,

\_id:1

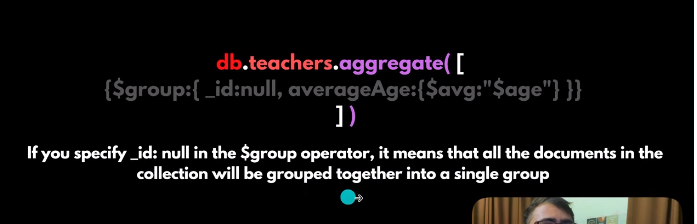
},

{

Hobbies: ‘’afafaf’,

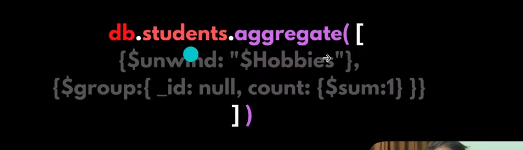
\_id:1

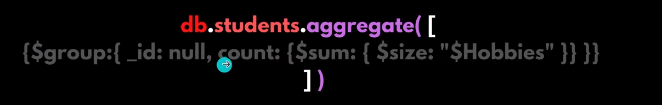
},



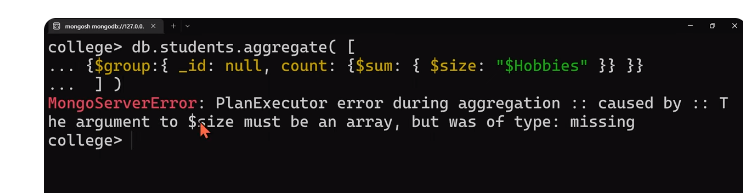


Two ways

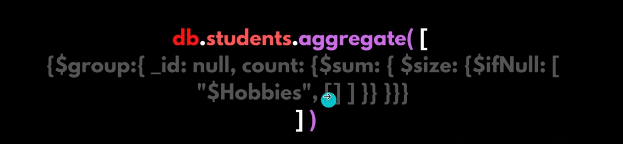




In second method if anyone not have Hobbies field we see error



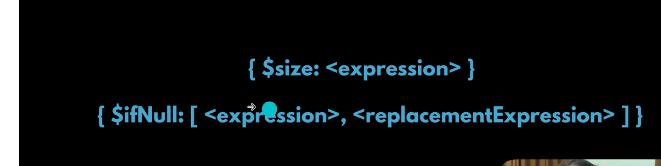
To solve above error



  const data =await model1.aggregate([{$group:{\_id:null,count:{$sum:{$size:{$ifNull:[

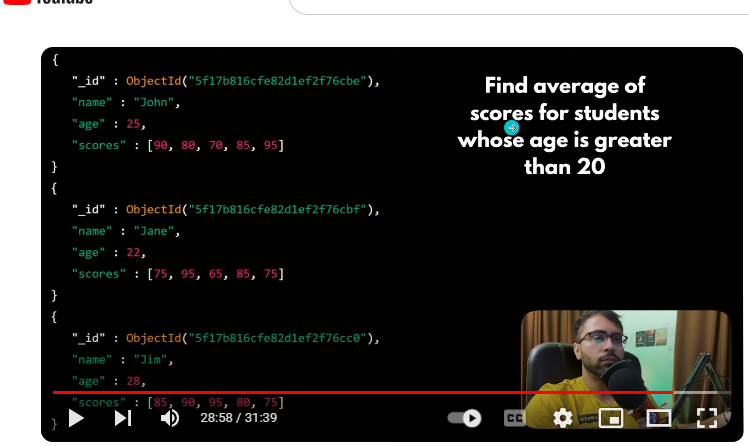
    "$Hobbies",[]

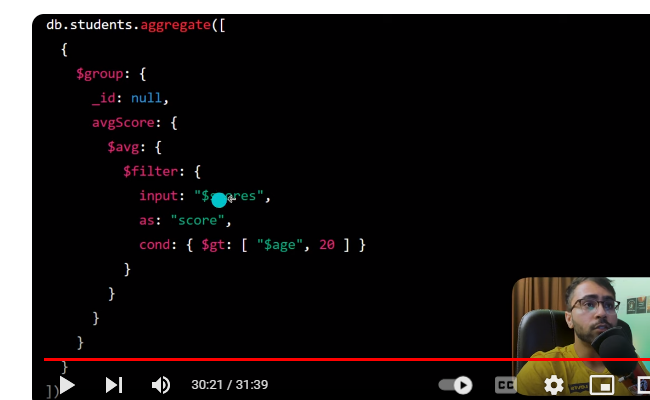
  ]}}}}}])

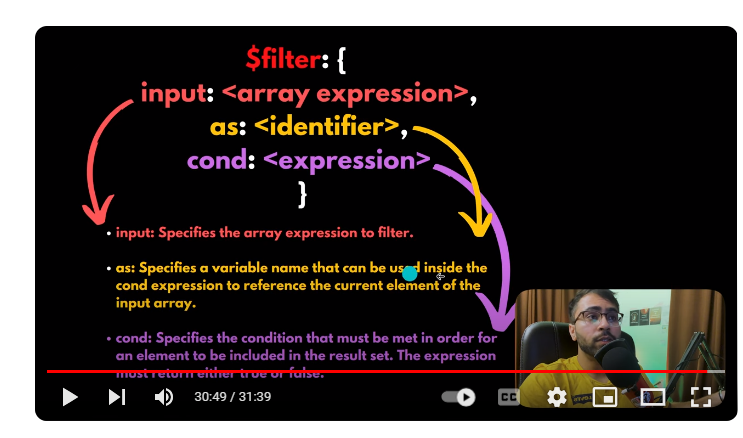


**$Filter**









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

    try {

//get only male teacher

    //   const data =await teacherModel.aggregate([{$match:{gender:'female'}}])

      //group by  age

    //   const data =await teacherModel.aggregate([{$group:{\_id:'$age'}}])

      //group by age wirh names

    //   const data =await teacherModel.aggregate([{$group:{\_id:'$age',names:{$push:"$name"}}}])

    //group by age with complete document

    // const data =await teacherModel.aggregate([    //give a count per age of male teacher

//givr  a count per age of male teacher

    // const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}}])

  //give a count per age of male student and sort them by count in desc order

  // const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}},{$sort:{countOfTeacherIngroup:-1}}])

  //max of sort teacher

    // const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$group:{\_id:'$age',countOfTeacherIngroup:{$sum:1}}},{$sort:{countOfTeacherIngroup:-1}},{$group:{\_id:null,maxNumberinAgeGroup:{$max:'$countOfTeacherIngroup'}}}])

//fech student with age group with hobbies

    // const data =await model1.aggregate([{$group:{\_id:"$age",hobbies:{$push:"$Hobbies"}}},])

  // using $unwind

  // const data =await model1.aggregate([{$unwind:"$Hobbies"},{$group:{\_id:"$age",hobbies:{$push:"$Hobbies"}}},])

  //average age

  // const data =await model1.aggregate([{$group:{\_id:null,averageAge:{$avg:"$age"}}},])

  //find total no of hobbies for all the students

  //first way

  // const data =await model1.aggregate([{$unwind:"$Hobbies"},{$group:{\_id:null,count:{$sum:1}}}])

//second way

// we use ifNull if some one not have field hobbies

  // const data =await model1.aggregate([{$group:{\_id:null,count:{$sum:{$size:{$ifNull:[

  //   "$Hobbies",[]

  // ]}}}}}])

  //get hobies without repetation

  // const data =await model1.aggregate([{$unwind:"$Hobbies"},{$group:{\_id:null,allhabbits:{$addToSet:"$Hobbies"}}}])

  //find avg of scores of student whose age is greater then 20

  // its not workong

  const data =await scoreModel.aggregate([{$group:{\_id:null,avgScore:{$avg:{$filter:{

    input:"$score",

    as :"scores",

    cond: {$gt:["$age",18]}

  }}}}}])

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

    } catch (error) {

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

    }

  }

  addScore =async(req,res)=>{

    try {

      var d =await scoreModel({

        name:"Tim",

        age:18,

        score:[85,60,95,80,75]

      })

    var data =  await d.save()

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

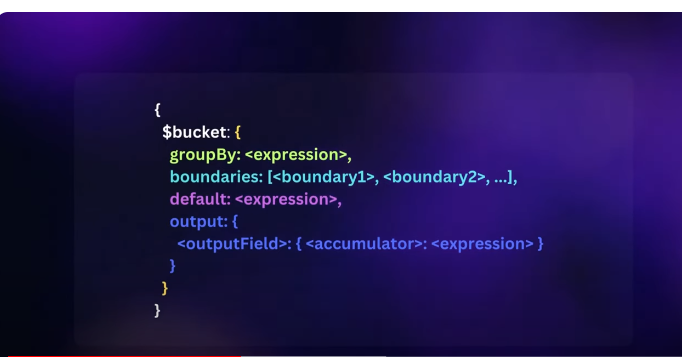
    } catch (error) {

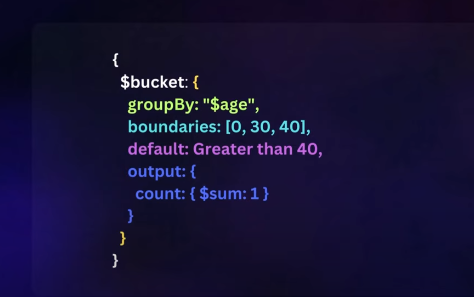
      return  res.status(400).json({ success:false,error:error})

    }

  }

**$Bucket**

****

****

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

    try {

  const data =await teacherModel.aggregate([{$match:{gender:"male"}},{$bucket:{

    groupBy:"$age",boundaries:[0,30,40],default:"Greater then 40 wala group",

output:{count:{$sum:1},names:{$push:"$name"}}

  }}])

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

    } catch (error) {

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

    }

  }

# $lookup in MongoDB

****

  // 65298919fcc9d9d67d3ef0ed

  // 652989320af897c69875e7fc

  // 652989406cab6eeb52d1eda7

//create order with hardcodeind tecaher id

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

  try {

const d =await order({

  order\_number:4,

  customer\_id:"652989406cab6eeb52d1eda7"

})

// const data =await teacherModel.find()

const data  =await d.save()

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

  } catch (error) {

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

  }

}

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

  try {

    //left outer join

const data =await teacherModel.aggregate([{$lookup:{from:"orders",localField:"\_id",foreignField:"customer\_id",as:"orderDetails"}},])

// const data =await order.find().populate('customer\_id')

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

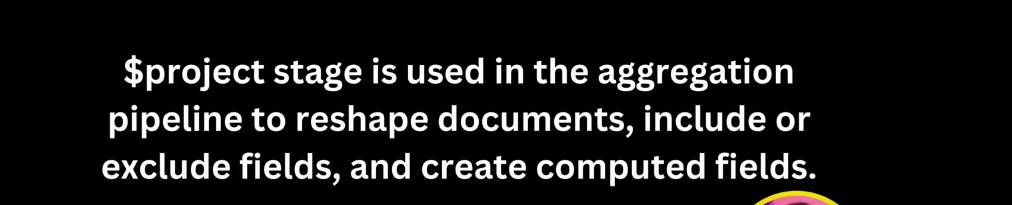
  } catch (error) {

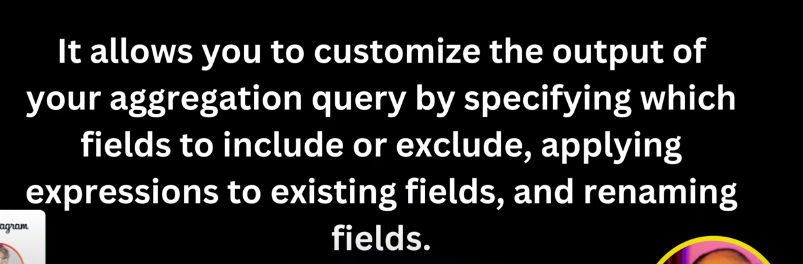
     return  res.status(400).json({ success:false,error:error})

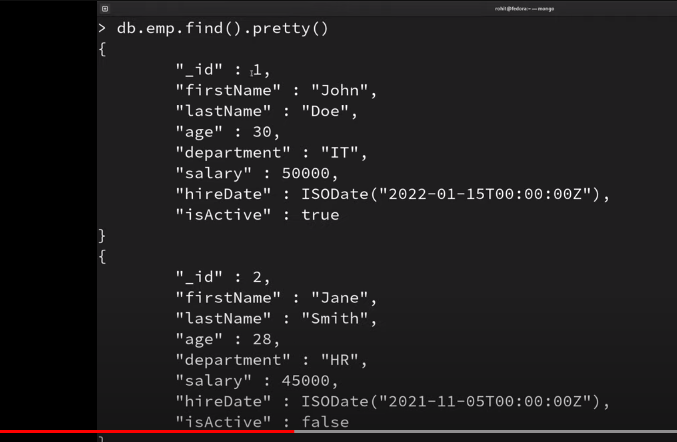
  }

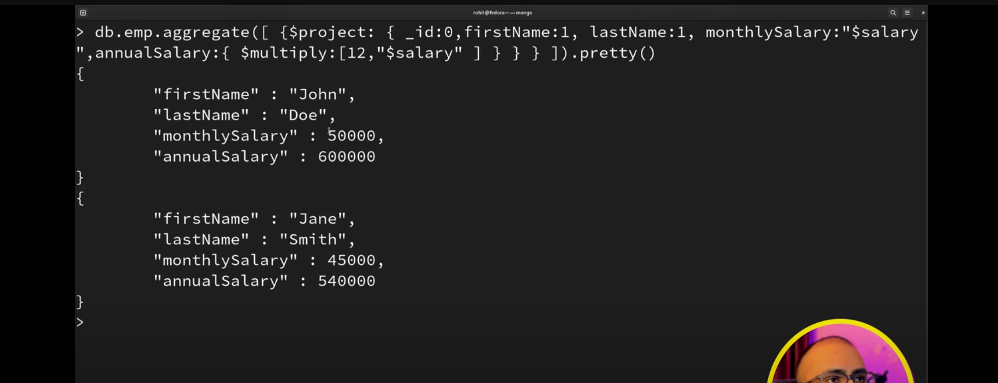
}

# $project in MongoDB

****

****

****

****

//use project

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

  try {

// use $project

// const data =await teacherModel.aggregate([{$project:{\_id:0}}])

//rename username

const data =await teacherModel.aggregate([{$project:{\_id:0,"userName":"$name"}}])

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

  } catch (error) {

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

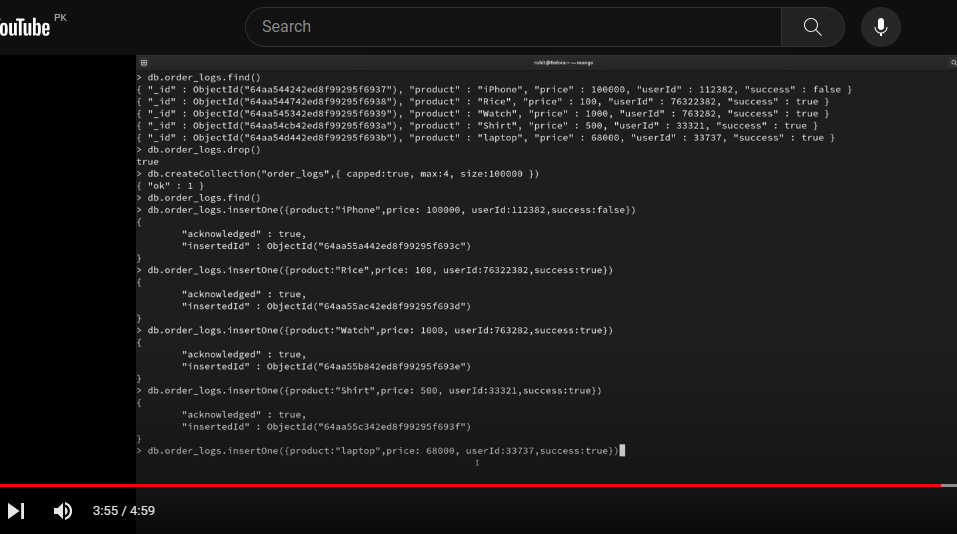
  }

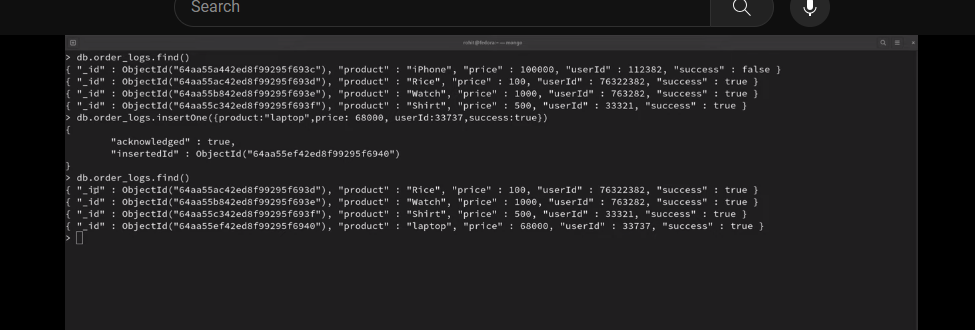
}

# Capped Collection in MongoDB

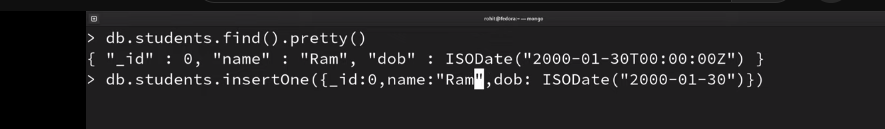
# In capt collection we define max document size and also space

# In when max doc size is greater then given size 1 doc delected automatically

****

****

# Mastering Date Queries in MongoDB: $year, $month, $dateToString, and More

****

****

**Add two hours in utc +20:200**

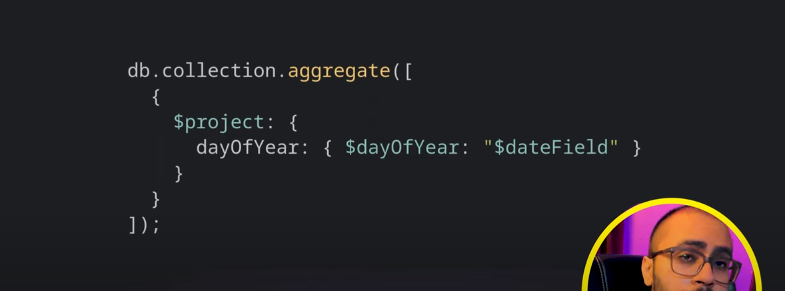
****

****

****

****

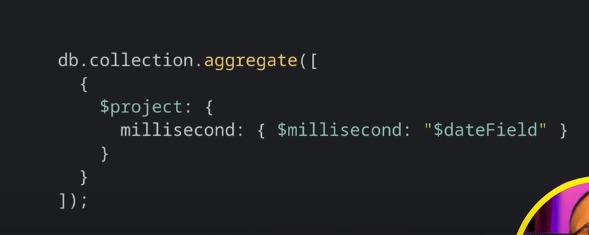
****

****

****

****

****

****

****

****

addBirthday =async(req,res)=>{

    try {

      var d =await birthDayModel({

        name:"zara",

    //   dob:new Date("2003-01-25T10:20:23Z+02:00").toISOString()

    //for current date

    dob:new Date().toISOString()

      //change value in uct for two hours  +02:00   its not working ?

    //   dob:new Date("2005-01-25T10:20:23Z++02:00").toISOString()

      })

    var data =  await d.save()

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

    } catch (error) {

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

    }

  }

  queryOnDate= async(req,res)=>{

try {

    // var data =  await birthDayModel.find().sort({dob:-1})

    //group doc with year

    // var data =  await birthDayModel.aggregate([{$group:{\_id:{$year :"$dob"},names:{$push:"$name"}}}])

    //get day of month

    // var data =  await birthDayModel.aggregate([{$match:{name:"Tim"}}, {$project:{dayOfMonth:{$dayOfMonth:"$dob"}}}])

    //get day year month

//     var data =  await birthDayModel.aggregate([{$match:{name:"zara"}}, {$project:{dayOfMonth:{$dayOfMonth:"$dob"}

// ,month:{$month:"$dob"},

// dayOfYear:{$dayOfYear:"$dob"},

// hour:{$hour:"$dob"},

// ms:{$millisecond:"$dob"}

// }}])

var data =  await birthDayModel.aggregate([{$project:{name:1,\_id:0,dob:{$dateToString:{format:"%d/%m/%Y %H:%M:%S", date:"$dob"}}}}])

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

} catch (error) {

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

}

  }