**Aggregate Usage**

**$match stage** – used to filter data

db.contacts.aggregate([ {$match : {gender : “female”}} ])

**$group stage** – used to group data on certain fields

db.contacts.aggregate([

    { $match: { gender: "female" } },

    { $group: { \_id: { state: "$location.state" }, totalPersons: {$sum : 1} }}

]).pretty()

Here in group the **name** of **state** will be shown as {state} and totalPersons will sum up the number of persons belongs to the state

**$sort stage** : this can be done only after the group stage and will sort data that is passed by group stage

db.contacts.aggregate([

    { $match: { gender: "female" } },

    { $group: { \_id: { state: "$location.state" }, totalPersons: { $sum: 1 } } },

    { $sort: {totalPersons : -1} }

]).pretty()

Also for calculating according to age and average age per gender

db.contacts.aggregate([

    { $match: { "dob.age": {$gt: 50} } },

    {

        $group:

        {

            \_id: { gender: "$gender" },

            totalPersons: { $sum: 1 },

            averageAge: { $avg: "$dob.age" }

        }

    },

    { $sort: {totalPersons : -1} }

]).pretty()

**$project stage –** it transforms data according to our need or what we want to see or include in our result from the database . Ex –

db.contacts.aggregate([

    {$project : {\_id : 0, gender: "$gender", fullName: {$concat: ["$name.first", " ", "$name.last"]}}}

]).pretty()

Here **$concat** will merge data first and last name to fullName

Below code for First and last name with first letter upper case

db.contacts.aggregate([

{

    $project: {

        \_id: 0,

        gender: "$gender",

        fullName: {

            $concat: [

                { $toUpper: { $substrCP: ["$name.first", 0, 1] } },

                {

                    $substrCP: ["$name.first", 1, { $subtract: [{ $strLenCP: "$name.first" }, 1] }

                    ]

                },

                " ",

                { $toUpper: { $substrCP: ["$name.last", 0, 1] } },

                {

                    $substrCP: ["$name.last", 1, { $subtract: [{ $strLenCP: "$name.last" }, 1] }

                    ]

                }

            ]

        }

    }

}

]).pretty()

**Note**  - We can have same stages multiple times, but we have to include transformed fields from the previous stage to next stage , that are required in results. For EX- in the code below in the first project, **location field is passed to next project and name : 1, email: 1,** because it is required in the next stage . Here we are also including geoJSON object in our result

db.contacts.aggregate([

    {

        $project: {

            \_id: 0,

            name: 1,

            email: 1,

            location: {

                type: "Point",

                coordinates: ["$location.coordinates.longitude", "$location.coordinates.latitude"]

            }

        }

    },

{

    $project: {

        \_id: 0,

        gender: "$gender",

email : 1,

        location: 1,

        fullName: {

            $concat: [

                { $toUpper: { $substrCP: ["$name.first", 0, 1] } },

                {

                    $substrCP: ["$name.first", 1, { $subtract: [{ $strLenCP: "$name.first" }, 1] }

                    ]

                },

                " ",

                { $toUpper: { $substrCP: ["$name.last", 0, 1] } },

                {

                    $substrCP: ["$name.last", 1, { $subtract: [{ $strLenCP: "$name.last" }, 1] }

                    ]

                }

            ]

        }

    }

}

]).pretty()

For converting string to numbers we can do as follows-

$convert is used to do so.

db.contacts.aggregate([

                coordinates: [

                    {

                        $convert:

                        {

                            input: "$location.coordinates.longitude",

                            to: "double",

                            onError: 0.0,

                            onNull: 0.0

                        }

                    },

                    {

                        $convert:

                        {

                            input: "$location.coordinates.latitude",

                            to: "double",

                            onError: 0.0,

                            onNull: 0.0

                        }

                    }

    }

}

]).pretty()

Transforming birthdate we can do as follows;

birthdate: { $convert: { input: "$dob.date", to: "date" } },

Extracting year from date

{ birthyear: { $isoWeekYear: "$birthdate" } }

**Pushing elements to newly created arrays**

db.friends.aggregate([

    { $group: { \_id: { age: "$age" }, allHobbies: { $push: "$hobbies" }}}

])

**$uswind stage-** this is used to pull out elements from an array, But it splits the documents for every elements of array to a single documents. Like

db.friends.aggregate([

    {$unwind: "$hobbies"}

])

To group all the elements in a single array we can do as

db.friends.aggregate([

{$unwind: "$hobbies"},

    { $group: { \_id: { age: "$age" }, allHobbies: { $push: "$hobbies" }}}

])

**Note- But the above code give the duplicate values too. To remove duplicates we can use addToSet in place of push as**

db.friends.aggregate([

{$unwind: "$hobbies"},

    { $group: { \_id: { age: "$age" }, allHobbies: { $addToSet: "$hobbies" }}}

])

If we want to pull out certain elements from objects in an array we can also do it using **$project and $slice operator**. Ex

db.friends.aggregate([

    {$project: {\_id : 0, examScore: {$slice: ["$examScores",1 ]}}}

])

**Using $filter operator.**

db.friends.aggregate([

    {

        $project: {

            \_id: 0,

            scores: {$filter: {input: "$examScores" ,as : "sc", cond: {$gt: ["$$sc.score", 60]}}}

        }

    }

])