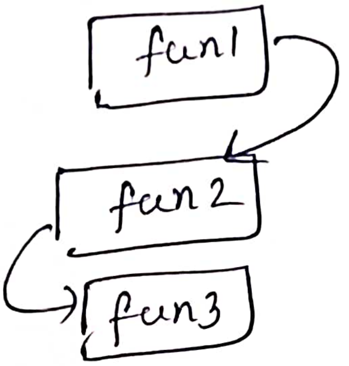
MongoDB Aggregation

Aggregation ⇒ To write aggregate query
⇓
A pipeline operation



one query result will be input to next query
∴ sequential operation

It groups the data from multiple documents into a single document based on specified expression

The aggregation process in MongoDB consists of several stages, each stage transforms the data in some way.
The output of one stage is fed as the input to next stage and so on. until the final stage produces the desired result.

MongoDB provides several built in aggregation pipeline stages to perform various operations on the data, such as $group, $sum, $avg, $min, $max, etc.

How to achieve this pipeline in mongodb

argument

```
>db.<collection name>.aggregate(pipeline,
                                options)
```

pipeline :- array of different operations

```
>db.~~teachers~~ students.aggregate([{$match:
                                      {gender
                                       :"male"
                                      }}])
```

we can use find also

Group ~~by~~ student by age

_id:- based on this you group

```
>db.students.aggregate([{$group:
                          {_id:"$age"}}
                       ])
```

It will group documents by age field, creating a new document for each unique age value.

The _id field in the group stage specifies the field based on which the documents will be grouped.

```
{_id: 21} ⇐ group with age 21
{_id: 20}
{_id: 19}
```

```
$group : { _id : expressim,
           field 1 : expression,
           field 2 : expressim, -- }
```

Group students by age and show all students' names per age group

```
> db.students.aggregate([
    {$group : {_id : "$age",
               names : {$push : "$name"
                       }}}
  ])
```

Ans

```
[
  {_id : 21} , names : ['ABC', 'XYZ', --]
  {_id : 20,   names : [                ]

]
```

The names field uses the $push operator to add the name field from each document in the group to an array

```
> db.students.aggregate([{group:
                          {_id: "$age",
                          ~~names:~~
                          ~~addtoc~~
                          students:
                          {$push:
                             "$$ROOT'
                                 }}}
                          ])
```

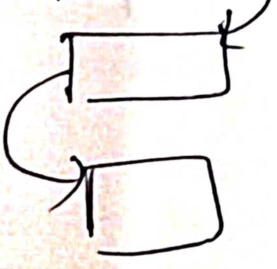
```
students:
[{name:   ,  }
   {      }
  }
```

$$ROOT represents complete document

Give a count per age of female students
⇓
group

```
>db.students.aggregate([
     {$match: {gender: "female"}},
    {$group: {_id: "$age", numberofStude
                         : {$sum: 1
                            }}}
                         ])
```

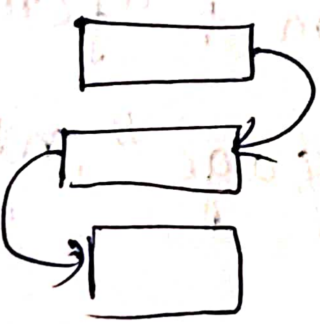
```
{_id: 21,
numberofStudent: ~~21~~ 2
}
{_id: 19
```



The value of $sum is 1, which means that for each document in the group, the value of anumber" will be incremented by 1

Q) Give a count per age of female students and sort them by count in desc manner.

```
>db.student.aggregate([
    {$match : {gender : "female"
                                   }},
    {$group : {_id : "$age",
                 numberofStudents : {$sum : 1
                                          }}}
    {$sort : {number of students : -1}
                                   }
                          ])
```



We can create pipeline as much as possible