Aggregation

Open windows prompt and connect to mongos in a sharded cluster or to any other standalone mongo database

```
mongo --port 27200

mongos> use cars
switched to db cars

mongos> db.list.find({},{_id:0})
{ "brand" : "Toyota", "model" : "Camry", "engine" : "Petrol", "gearbox" : "Auto", "mileage" : 120000, "year" : 2010 }
{ "brand" : "Honda", "model" : "Accord", "engine" : "Diesel", "gearbox" : "CVT", "mileage" : NumberLong(65000), "year" : 2010 }
{ "brand" : "Ford", "model" : "F-150", "engine" : "Petrol", "gearbox" : "Auto", "mileage" : 27000, "year" : 2015 }
{ "brand" : "Nissan", "model" : "Leaf", "engine" : "Electric", "gearbox" : "CVT", "mileage" : 15000, "year" : 2010 }
{ "brand" : "Honda", "model" : "Civic", "engine" : "Diesel", "gearbox" : "Manual", "mileage" : 35000, "year" : 2016 }
{ "brand" : "Honda", "model" : "CR-V", "engine" : "Hybrid", "gearbox" : "CVT" }
{ "brand" : "Toyota", "model" : "Corolla", "engine" : "Petrol", "gearbox" : "Manual", "mileage" : NumberLong(200000), "year" : 2000 }
{ "brand" : "Lexus", "model" : "RX350H", "engine" : "Hybrid", "gearbox" : "CVT", "mileage" : 7000, "year" : 2018 }
{ "brand" : "Toyota", "model" : "Supra", "engine" : "Petrol", "gearbox" : "Manual" }
{ "brand" : "Honda", "model" : "Civic", "engine" : "Petrol", "gearbox" : "Auto", "mileage" : 5000, "year" : 2019 }

mongos> db.list.count()
10

mongos> db.list.count({brand:'Toyota'})
3

mongos> db.list.distinct('brand')
[ "Ford", "Honda", "Lexus", "Nissan", "Toyota" ]

mongos> db.list.distinct('engine')
[ "Diesel", "Electric", "Hybrid", "Petrol" ]
mongos> db.list.distinct('engine',{brand:{$in:['Toyota','Lexus']}})
[ "Hybrid", "Petrol" ]
```

```
mongos> load('pipeline.js')
true

mongos>  sales_record
[
	{
		"staff_id" : 101,
		"name" : "Tom",
		"sales" : 6000,
		"month" : 1
	},
	{
		"staff_id" : 202,
		"name" : "Jerry",
		"sales" : 5000,
		"month" : 1
	},
	{
		"staff_id" : 101,
		"name" : "Tom",
		"sales" : 4000,
		"month" : 2
	},
	{
		"staff_id" : 202,
		"name" : "Jerry",
		"sales" : 7000,
		"month" : 2
	},
	{
		"staff_id" : 101,
		"name" : "Tom",
		"sales" : 5000,
		"month" : 3
	},
	{
		"staff_id" : 202,
		"name" : "Jerry",
		"sales" : 9000,
		"month" : 3
	}
]
```

```
mongos> db.sales.insert(sales_record)
BulkWriteResult({
	"writeErrors" : [ ],
	"writeConcernErrors" : [ ],
	"nInserted" : 6,
	"nUpserted" : 0,
	"nMatched" : 0,
	"nModified" : 0,
	"nRemoved" : 0,
	"upserted" : [ ]
})

mongos> db.sales.find()
{ "_id" : ObjectId("60b11795d280bd9dbdc975c9"), "staff_id" : 101, "name" : "Tom", "sales" :
6000, "month" : 1 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975ca"), "staff_id" : 202, "name" : "Jerry", "sales" :
5000, "month" : 1 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975cb"), "staff_id" : 101, "name" : "Tom", "sales" :
4000, "month" : 2 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975cc"), "staff_id" : 202, "name" : "Jerry", "sales" :
7000, "month" : 2 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975cd"), "staff_id" : 101, "name" : "Tom", "sales" :
5000, "month" : 3 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975ce"), "staff_id" : 202, "name" : "Jerry", "sales" :
9000, "month" : 3 }


mongos> db.sales.aggregate({$match:{name:{$eq:'Tom'}}})
{ "_id" : ObjectId("60b11795d280bd9dbdc975c9"), "staff_id" : 101, "name" : "Tom", "sales" :
6000, "month" : 1 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975cb"), "staff_id" : 101, "name" : "Tom", "sales" :
4000, "month" : 2 }
{ "_id" : ObjectId("60b11795d280bd9dbdc975cd"), "staff_id" : 101, "name" : "Tom", "sales" :
5000, "month" : 3 }

mongos> db.sales.aggregate({$match:{name:{$eq:'Tom'}}},{$project:{_id:0,sales:1}})
{ "sales" : 6000 }
{ "sales" : 4000 }
{ "sales" : 5000 }
```

```
mongos>
db.sales.aggregate({$match:{name:{$eq:'Tom'}}},{$project:{_id:0,sales:1}},{$group:{_id:'Tom',to
tal_sales:{$sum:'$sales'}}})
{ "_id" : "Tom", "total_sales" : 15000 }


mongos>
db.sales.aggregate({$match:{name:{$eq:'Tom'}}},{$project:{_id:0,sales:1}},{$group:{_id:'Tom',to
tal_sales:{$sum:'$sales'},avg_sales:{$avg:'$sales'}}})
{ "_id" : "Tom", "total_sales" : 15000, "avg_sales" : 5000 }

mongos> db.price.insert({_id:0,price:100,shipping:10,coupon:-5})
WriteResult({ "nInserted" : 1 })

mongos> db.price.aggregate({$project:{total_price:{$sum:['$price','$shipping','$coupon']}}})
{ "_id" : 0, "total_price" : 105 }

mongos> db.price.insert({_id:1,price:120,shipping:10,coupon:-5})
WriteResult({ "nInserted" : 1 })

mongos> db.price.aggregate({$group:{_id:'avg_price',avg_price:{$avg:'$price'}}})
{ "_id" : "avg_price", "avg_price" : 110 }

mongos> db.ratings.aggregate({$project:{total_rating:{$sum:'$ratings'}}})
{ "_id" : 0, "total_rating" : 42.2 }

mongos> db.ratings.aggregate({$project:{avg_rating:{$avg:'$ratings'}}})
{ "_id" : 0, "avg_rating" : 4.688888888888889 }

mongos> db.ratings.aggregate({$project:{ar:{$avg:'$ratings'}}},{$project:{tr:{$trunc:['$ar',2]}}})
{ "_id" : 0, "tr" : 4.68 }
```

```
mongos> db.products.insert({'name':'iPad 16GB Wifi', 'manufacturer':"Apple",
                            'category':'Tablets',
                            'price':499.00})
mongos> db.products.insert({'name':'iPad 32GB Wifi', 'category':'Tablets',
                            'manufacturer':"Apple",
                            'price':599.00})
mongos> db.products.insert({'name':'iPad 64GB Wifi', 'category':'Tablets',
                            'manufacturer':"Apple",
                            'price':699.00})
mongos> db.products.insert({'name':'Galaxy S3', 'category':'Cell Phones',
                            'manufacturer':'Samsung',
                            'price':563.99})
mongos> db.products.insert({'name':'Galaxy Tab 10', 'category':'Tablets',
                            'manufacturer':'Samsung',
                            'price':450.99})
mongos> db.products.insert({'name':'Vaio', 'category':'Laptops',
                            'manufacturer':"Sony",
                            'price':499.00})
mongos> db.products.insert({'name':'Macbook Air 13inch', 'category':'Laptops',
                            'manufacturer':"Apple",
                            'price':499.00})
mongos> db.products.insert({'name':'Nexus 7', 'category':'Tablets',
                            'manufacturer':"Google",
                            'price':199.00})
mongos> db.products.insert({'name':'Kindle Paper White', 'category':'Tablets',
                            'manufacturer':"Amazon",
                            'price':129.00})
mongos> db.products.insert({'name':'Kindle Fire', 'category':'Tablets',
                            'manufacturer':"Amazon",
                            'price':199.00})


/****First aggregate Querry****Count of each manufacturer********/
mongos> db.products.aggregate([
      {$group:
       {
         _id:"$manufacturer",
         num_products:{$sum:1}
       }
      }
])
```

/***Agrregation Pipeline*****************/

Collection->$project->$match->$group->$sort->Result

$project: reshape the document
$match: Filter step
$group: aggregate(sum, avg,count etc)
$sort: sorting the document
$skip: Skip documents
$limit: limit number of document
$unwind: Normalize tag /*
tags;["red","blue","green"]
tag:"red",
tag:"blue",
tag:"green" */

$out: output

/***************Count of manufacturer with category/

```
mongos> db.products.aggregate([
      {$group:
       {
          _id:{ "manufacturer":"$manufacturer",
               "category":"$category"
              },
          num_products:{$sum:1}
       }
      }
])
```

```
/*** Id can have values***************/

mongos> db.foo.insert({_id:{name:"aakash",class:"7module"}, hometown:"IND"})

/****using sum operator **********/

mongos> db.products.aggregate([
      {$group:
       {
         _id:{"manufacturer":"$manufacturer"},
         total_price:{$sum:"$price"}
       }
      }
])



/****Using avg function  ****************/

mongos> db.products.aggregate([
      {$group:
       {
         _id:{"manufacturer":"$manufacturer"},
         avg_price:{$avg:"$price"}
       }
      }
])



/****Addto set ****************/
mongos> db.products.aggregate([
      {$group:
       {
         _id:{"manufacturer":"$manufacturer"},
         categories:{$addToSet:"$category"}
       }
      }
])
```

```
/***********Push function **************/
mongos> db.products.aggregate([
      {$group:
       {
         _id:{"manufacturer":"$manufacturer"},
         categories:{$push:"$category"}
       }
      }
])


/***********Max function **************/
mongos> db.products.aggregate([
      {$group:
       {
         _id:{"manufacturer":"$manufacturer"},
         maxPrice:{$max:"$price"}
       }
      }
])


/**Double group***************/
mongos> db.grades.aggregate([
  {'$group':{_id:{class_id:"$class_id", student_id:"$student_id"}, 'average':{"$avg":"$score"}}},
  {'$group':{_id:"$_id.class_id", 'average':{"$avg":"$average"}}}])



/**********$project *********************/
mongos> db.products.aggregate([
      {$project:
        {
          _id:1,
          'maker':{$toLower: "$manufacturer"},
          'details':{'category':"$category",
                     'price':{"$multiply":["$price",10]}
                    },
           'item':'$name'
        }
      }
])
```

```
 mongos> db.city.count({city:"CHICOPEE"})
mongos> db.city.distinct("state")
 mongos> db.city.distinct("city").length
/***********$match***************/
mongos> db.city.aggregate([
   {$match:
     {
       state:"CA"
     }
   }

])




/**************$match with $group********/
mongos> db.city.aggregate([
   {$match:
     {
       state:"CA"
     }
   },
   {$group:
     {
       _id: "$city",
       population: {$sum: "$pop"},
       zip_codes: {$addToSet: "$_id"}
     }
   }
])
```

/**************$match with $group and $project********/

```
mongos> db.city.aggregate([
   {$match:
    {
      state:"CA"
    }
   },
   {$group:
    {
      _id: "$city",
      population: {$sum: "$pop"},
      zip_codes: {$addToSet: "$_id"}
    }
   },
    {$project:
    {
      _id: 0,
      city: "$_id",
      population: 1,
      zip_codes: 1
    }
   }

])
```

/**************$match with $group and $project add sort to this********/

```
mongos> db.city.aggregate([
   {$match:
    {
      state:"CA"
    }
   },
   {$group:
    {
      _id: "$city",
      population: {$sum: "$pop"},
      zip_codes: {$addToSet: "$_id"}
    }
   },
    {$project:
    {
      _id: 0,
      city: "$_id",
      population: 1,
      zip_codes: 1
    }
   },
    {$sort:
    {
      population: 1
    }
   },
   {$skip: 10},
   {$limit: 5}

])
```

```
/**********************Map reduce *************************/
mongos> db.collection.mapReduce(
                    function(){emit(key,value)},
                    function(key, values){return reduceFunction},
                       { out: collection,
                         query:document,
                          sort: document,
                         limit:number
                        }
                      )


mongos> db.books.insert({name:"JAVA", pages:100});

mongos> db.books.insert({name:"XML", pages:400});

mongos> db.books.insert({name:"JSON", pages:300});

mongos> db.books.insert({name:"AngularJs", pages:250});

mongos> db.books.insert({name:"NodeJS", pages:150});


var map = function(){
var category;
if(this.pages >=250)
 {
   category='Big Books';
 }
else
    category ="small Book";
    emit(category, {name:this.name})

};



var reduce = function(key, values){
  var sum = 0;
  values.forEach(function(doc){
  sum += 1;
  });
return {books:sum};
};
```

var count = mongos> db.books.mapReduce(map, reduce, {out: "book_result"});

/*************Order of skip sort and limit************/
$sort -> $skip -> $limit

mongos> db.restaurants.getShardDistribution()

Shard shard2rs at shard2rs/localhost:2004,localhost:2005,localhost:2006
 data : 57.64MiB docs : 333202 chunks : 2
 estimated data per chunk : 28.82MiB
 estimated docs per chunk : 166601

Shard shard4rs at shard4rs/localhost:20015,localhost:20016,localhost:20017
 data : 57.77MiB docs : 333955 chunks : 2
 estimated data per chunk : 28.88MiB
 estimated docs per chunk : 166977

Shard shard3rs at shard3rs/localhost:2007,localhost:2008,localhost:2009
 data : 57.6MiB docs : 332843 chunks : 2
 estimated data per chunk : 28.8MiB
 estimated docs per chunk : 166421

Totals
 data : 173.02MiB docs : 1000000 chunks : 6
 Shard shard2rs contains 33.31% data, 33.32% docs in cluster, avg obj size on shard : 181B
 Shard shard4rs contains 33.39% data, 33.39% docs in cluster, avg obj size on shard : 181B
 Shard shard3rs contains 33.29% data, 33.28% docs in cluster, avg obj size on shard : 181B

```
mongos> db.restaurants.aggregate([
...   {
...     "$project": {
...       "avgRating": { "$sum": "$reviews.rating" }
...     }
...   },
...   {
...     "$out": "restaurants"
...   }
... ])
uncaught exception: Error: command failed: {
        "ok" : 0,
        "errmsg" : "mongoMart.restaurants cannot be sharded",
        "code" : 28769,
        "codeName" : "Location28769",
        "operationTime" : Timestamp(1624975122, 2),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1624975122, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
} : aggregate failed
```