

Aggregation

লার্জ স্কেলের ডাটা ম্যানেজ করার জন্য **Aggregation** ব্যবহার করা হয়। **Aggregation** স্টেজ পাইপ লাইন ব্যবহার করে কাজ করে। প্রতিটা স্টেজ হচ্ছে একটি করে পাইপলাইন যার মধ্যে আলাদা আলাদা ভাবে কন্ডিশন ব্যবহার করা যায়। আমরা চাইলে একাধিক **\$match** অপারেটর ব্যবহার করতে পারি কিন্তু যত বেশি স্টেজ ব্যবহার করা হবে তত বেশি প্রসেসিং টাইম নেবে।

প্রথম স্টেজে **\$match** ব্যবহার করে কন্ডিশনের উপরে ভিত্তি করে ডাটা নিয়ে আসা

এই ডাটার মধ্যে যদি কোন মডিফাই করার প্রয়োজন পড়ে যা **অরজিনাল ডাটাকে মডিফাই করবে না** শুধুমাত্র আউটপুট ডাটাকে মডিফাই করে একটা রেজাল্ট প্রদান করবে, তাহলে **\$addFields** ইউজ করতে পারি এবং পরবর্তী স্টেজে যদি আমরা সেই রেজাল্টকে অরজিনাল ডকুমেন্টে কোন পরিবর্তন না করে আলাদা একটি **কালেকশনে** পুশ করতে চাই তাহলে **\$out** ইউজ করব

আর যদি অরজিনাল ডকুমেন্ট কে মডিফাই করতে চাই তাহলে **\$merge** করব

সামারি হচ্ছে এপ্লিকেশনে

\$match

সাহায্যে ডাটা ফিল্টার করে পরবর্তী স্টেজ ডাটা পাঠানো হয় এবং সেখানে

\$addFields

ব্যবহার করে পাইপ লাইনের ডকুমেন্ট কে মডিফাই করা হয় যা অরজিনাল ডাটায় কোন প্রভাব পড়বে না

Example:


```
db.data.aggregate([
  // stage-1
  { $match: { gender: "Male" } },
  // stage-2 -- insert extra fields without modify original field
  { $addFields: { year: 2023, course: { name: "Next level", batch: 2 } } },
  //stage-3
  { $project: { gender: 1, course: 1, year: 1 } },
  // stage-4
  { $out: "output-collection" },
  // stage-5 -- It modify original data
  // { $merge: "data" }
])
```

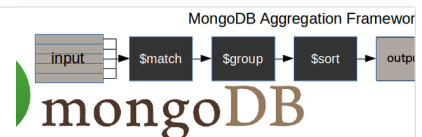
▼ **\$group**

Source:

MongoDB Aggregation: tutorial with examples and exercises | Studio 3T

Master the MongoDB aggregation pipeline. Follow along with query examples using the most important aggregation stages, and test your skills!

 <https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/#mongodb-group>



\$group ব্যবহার করা হয় একই ক্যাটাগরির ডাটা কে আলাদাভাবে ক্যাটাগরির আন্ডারে গ্রুপ করে রাখার জন্য।

যেমন ১০০ ইউজারের মধ্যে পুরুষ এবং নারী দুইটা গ্রুপ তৈরি করা যায়। আবার স্টুডেন্ট হলে ক্লাস ভিত্তিক গ্রুপ করা যায় আবার একই ক্লাসের মধ্যে সেকশন হিসেবে গ্রুপ করা যায়।

গ্রুপ করার জন্য দরকার একটি **প্রাইমারি কি** যার উপর ভিত্তি করে গ্রুপ করা হবে, তাকে **_id:** দ্বারা প্রকাশ করা হয় example: **_id: "\$address.country"**

প্রতি গ্রুপে কতগুলো আছে সেটি কাউন্ট করার জন্য **count: { \$sum: 1 }**

ব্যবহার করা হয়. NT: এখানে আমরা count, max, min, avg, sum, push ব্যবহার করতে পারি

- চাইলে গ্রুপকৃত ডকুমেন্ট গুলোর সবগুলো ফিল্ড পাঠানো যায় এর জন্য **fullDoc: { \$push: "\$\$ROOT" }** ব্যবহার করতে হয়.

▼ Example:

```
db.data.aggregate([
  // stage-1
  {
    $group: {
      _id: "$gender",
      count: { $sum: 1 },
      fullDocs: { $push: "$$ROOT" }
    }
  }
])
```

- আমরা যদি সবগুলো ফিল্ড না পাঠিয়ে স্পেসিফিক কতগুলো ফিল্ড পাঠাতে চাই তাহলে আরেকটি স্টেজ নিয়ে রেজাল্টের উপরে **\$project** করে ফিল্ড সিলেক্ট করে দিতে পারি.

▼ Example:

```
db.data.aggregate([
  // stage-1
  {
    $group: {
      _id: "$gender",
      count: { $sum: 1 },
      fullDocs: { $push: "$$ROOT" }
    }
  },
  // stage-2
  {
    $project: {
      "fullDocs.gender": 1,
      "fullDocs.phone": 1
    }
  }
])
```

▼ Output:

```
/* 1 */
{
  "_id" : "Genderfluid",
  "fullDocs" : [
    {
      "phone" : "(914) 6646925",
      "gender" : "Genderfluid"
    },
    {
      "phone" : "(990) 4889800",
      "gender" : "Genderfluid"
    },
    {
      "phone" : "(665) 6905129",
      "gender" : "Genderfluid"
    },
    {
      "phone" : "(388) 9242545",
      "gender" : "Genderfluid"
    }
  ]
},
```

- আর যদি কোন স্পেসিফিক একটি ফিল্ড প্রোভাইড করতে চাই তাহলে শুধুমাত্র **name: { \$push: "\$name" }** বসালে হবে।

▼ Example:

```
db.data.aggregate([
  // stage-1
  {
    $group: {
      _id: "$gender",
      count: { $sum: 1 },
      genderGroups: { $push: "$gender" }
    }
  }
])
```

▼ Output

```
/* 1 */
{
  "_id" : "Female",
  "count" : 52,
  "genderGroups" : [ "Female", "Female", ..... "Female", ],
/* 2 */
{
  "_id" : "Male",
  "count" : 38,
  "genderGroups" : [ "Male", "Male", ..... "Male", ],
/* 3 */
{
  "_id" : "Non-binary",
  "count" : 1,
  "genderGroups" : [ "Non-binary" ]
},
/* 4 */
{
  "_id" : "Genderfluid",
  "count" : 4,
  "genderGroups" : [ "Genderfluid", "Genderfluid", "Genderfluid", "Genderfluid" ]
},
```

▼ accumulator operator - (count, max, min, avg, sum, push) এর ব্যবহার

Source:

MongoDB Aggregation: tutorial with examples and exercises | Studio 3T

Master the MongoDB aggregation pipeline. Follow along with query examples using the most important aggregation stages, and test your skills!

<https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/#mongodb-group>

MongoDB Aggregation Framework

input → \$match → \$group → \$sort → output

mongoDB

টোটাল সব ডকুমেন্টের মধ্য থেকে যদি (count, max, min, avg, sum, push) বের করতে চাই তাহলে, আমাদেরকে সবগুলো ডকুমেন্টকে একটি গ্রুপের আন্ডারে নিয়ে আসতে হবে।

এর জন্য আমরা **_id** হিসেবে **null** ইউজ করব তাহলে, **null** গ্রুপের আন্ডারে সবগুলো ডকুমেন্ট চলে আসবে এরপর আমরা (count, max, min, avg, sum, push) এগুলো ব্যবহার করতে পারব।

Example:

```
// Input
db.data.aggregate([
  // stage-1
  {
    $group: {
      _id: "null",
      totalSalary: { $sum: "$salary" },
      averageSalary: { $avg: "$salary" },
      maxSalary: { $max: "$salary" },
      minSalary: { $min: "$salary" },
    }
  }
])
```

```
// Output
{
  "_id" : "null",
  "totalSalary" : 54037,
  "averageSalary" : 545.8282828282828,
  "maxSalary" : 24000,
  "minSalary" : 105
}
```

For specific.

▼ \$sum

```
// input
db.data.aggregate([
  // stage-1
  {
    $group: { _id: "null", totalSalary: { $sum: "$salary" } }
  }
])
```

```
// Output
{
  "_id" : "null",
  "totalSalary" : 54037
}
```

▼ \$avg

```
// Input
db.data.aggregate([
  // stage-1
  {
    $group: { _id: "null", averageSalary: { $avg: "$salary" } }
  }
])
```

```
// Output
{
  "_id" : "null",
  "averageSalary" : 545.8282828282828
}
```

Same as other operations.

▼ \$project

প্রজেক্ট স্টেজে শুধুমাত্র ফিল্ড ফিল্টারিং ছাড়াও ফিল্ডের নাম **Rename** করতে পারে এবং এক্সট্রা ফাংশনাল কাজ করতে পারে. like calculation

Example:

```
// Input
db.data.aggregate([
  // stage-1
  {
    $group: {
      _id: "null",
      totalSalary: { $sum: "$salary" },
      averageSalary: { $avg: "$salary" },
      maxSalary: { $max: "$salary" },
      minSalary: { $min: "$salary" },
    }
  },
  // stage-2
  {
    $project: {
      totalSalary: 1,
      maxSalary: 1,
      minSalary: 1,
      averageSalary: "$avgSalary",
      rangeBetweenManMin: { $subtract: [ "$maxSalary", "$minSalary" ] }
    }
  }
])
```

Output:

```
{
  "_id" : "null",
  "totalSalary" : 54037,
  "maxSalary" : 24000,
  "minSalary" : 105,
  "rangeBetweenManMin" : 23895
}
```

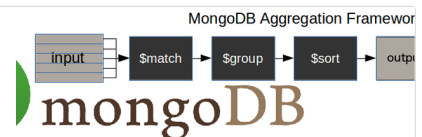
▼ \$unwind (এর ফিল্ডের ওপর গ্রুপ)

Source:

MongoDB Aggregation: tutorial with examples and exercises | Studio 3T

Master the MongoDB aggregation pipeline. Follow along with query examples using the most important aggregation stages, and test your skills!

<https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/#mongodb-unwind>



গ্রুপ ব্যবহার করে শুধুমাত্র যেকোনো একটা ফিল্ডের উপর ভিত্তি করে গ্রুপ করা যায়. যদি ওই ফিল্ডটি **Array** হয় তখন সেটাকে গ্রুপ করা যায় না. এর জন্য **\$unwind** ইউজ করতে হয়.

\$unwind প্রথমে প্রত্যেকটা ডিফারেন্ট **Array** এর ডাটা কে আলাদা করে গ্রুপ তৈরি করে, ধরা যাক একটি ফিল্ডের array তে পাঁচটি ডিফারেন্ট বেলু আছে **\$unwind** করলে পাঁচটি ডিফারেন্ট বেল এর জন্য পাঁচটি ডিফারেন্ট গ্রুপ তৈরি হবে এবং প্রতিটি গ্রুপ Main ফিল্ডের আইডিকে রেফার করবে. এখন চাইলে এই আলাদা ডিফারেন্ট গ্রুপগুলোকে নিয়ে **\$group** এপ্লাই করা যাবে তাহলে.

প্রথম স্টেজে ডাটা কে **\$unwind** করে সেকেন্ড স্টেজে সে আনওয়াইন্ড ডাটা গুলোকে **\$group** করবে.

Data format:

```
"interests" : [ "Gaming", "Writing", "Reading" ],
"interests" : [ "Gardening", "Travelling", "Writing" ],
```

\$unwind:

```
db.data.aggregate([
  // stage-1
  {
    $unwind: "$interests"
  }
])
```

\$unwind with \$group

```
// Input
db.data.aggregate([
  // stage-1
  {
    $unwind: "$interests"
  },
  {
    $group: { _id: "$interests", count: { $sum: 1 } }
  }
])
```


```
// Output
/* 1 */
{
  "_id" : "Gardening",
  "count" : 41
},
/* 2 */
{
  "_id" : "Writing",
  "count" : 54
},
/* 3 */
{
  "_id" : "Reading",
  "count" : 59
},
/* 4 */
{
  "_id" : "Travelling",
  "count" : 40
},
/* 5 */
{
  "_id" : "Cooking",
  "count" : 54
},
/* 6 */
{
  "_id" : "Gaming",
  "count" : 49
}
```

▼ \$bucket \$sort \$limit

Source:

\$bucket (aggregation)

.leafygreen-ui-zzkys8{font-size:16px;line-height:28px;font-family:'Euclid Circular A','Helvetica Neue',Helvetica,Arial,sans-serif;display:-webkit-inline-box;display:-webkit-inline-flex;display:-ms-inline-flexbox;display:inline-flex;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-

 <https://www.mongodb.com/docs/manual/reference/operator/aggregation/bucket/>



\$bucket হচ্ছে পাত্র, অনেকগুলো ডাটা কে কোন একটি ফিল্ডের বিভিন্ন বাউন্ডারি বা রেঞ্জের মাধ্যমে আলাদা আলাদা পাত্রে রাখতে পারি.

যেমন: যদি বয়সের কথা বলি তাহলে 10 থেকে 20 বছর একটি পাত্রে, ২০ থেকে ৩০, বছর আরেকটি পাত্রে এবং ৩০ থেকে ৪০ বছর আরও একটি পাত্রে. এভাবে আমরা পাত্র বানিয়ে ডাটা গুলো রাখতে পারি

groupBy হবে একটি ফিল্ড যার উপর ভিত্তি করে **\$bucket** তৈরি করা হবে

boundaries হবে একটি এর যার প্রতিটি বেলু হবে **\$bucket** বাউন্ডারি বা রেঞ্জের

default হিসেবে থাকবে উপরে বাউন্ডারির বাহিরে যে ডাটা গুলো থাকবে সেগুলো হবে ডিফল্ট ডাটা এবং **output** এর মাধ্যমে সবগুলো **\$bucket** কে একটি অবজেক্ট আকারে প্রদান করবে.

আমরা চাইলে **output** এর ভিতরে **accumulator operator** - (**count, max, min, avg, sum, push**) ব্যবহার করে ডাটা রেজাল্ট তৈরি করতে পারি.

Syntax:

```
{
  $bucket: {
    groupBy: <expression>,
    boundaries: [ <lowerbound1>, <lowerbound2>, ... ],
    default: <literal>,
    output: {
      <output1>: { <$accumulator expression> },
      ...
      <outputN>: { <$accumulator expression> }
    }
  }
}
```

Example:

```
// Input
db.data.aggregate([
  // stage-1
  {
    $bucket: {
      groupBy: "$age",
      boundaries: [20, 40, 60, 80],
      default: "others",
      output: {
        count: { $sum: 1 },
        // documents: { $push: "$$ROOT" }
      }
    }
  }
])
```

▼ Output:

```
// Output
/* 1 */
{
  "_id" : 20,
  "count" : 14
},

/* 2 */
{
  "_id" : 40,
  "count" : 18
},

/* 3 */
{
  "_id" : 60,
  "count" : 23
},

/* 4 */
{
  "_id" : "others",
  "count" : 44
}
```

Apply \$sort \$limit \$project

```
// Input
db.data.aggregate([
  // stage-1
  {
    $bucket: {
      groupBy: "$age",
      boundaries: [20, 40, 60, 80],
      default: "others",
      output: {
        count: { $sum: 1 },
        documents: { $push: "$$ROOT" }
      }
    }
  },
  // stage-2
  {
    $sort: { count: 1 }
  },
  // stage-3
  {
    $limit: 2
  },
  // stage-4
  {
    $project: {
      count: 1,
      documents: 1
    }
  }
])
```

▼ Output

```
/* 1 */
{
  "_id" : 20,
  "count" : 14,
  "documents" : [
    {
      "_id" : ObjectId("6406ad63fc13ae5a40000065"),
```



```

    "name" : {
      "firstName" : "Mariele",
      "lastName" : "Dangl"
    },
    "email" : "mdangl1@odnoklassniki.ru",
    "phone" : "(167) 7775715",
    "gender" : "Female",
    "age" : 21,
    "birthday" : "3/13/2022",
    "favoutiteColor" : "Aquamarine",
    "friends" : [
      "Najmus Sakib",
      "Mir Hussain",
      "Fahim Ahammed Firoz",
      "Tanmoy Parvez",
      "Abdur Rakib",
      "emdad"
    ],
    "occupation" : "Food Chemist",
    "interests" : [ "Cooking", "Writing", "Reading" ],
    "salary" : 24000
  },
  .....
]
},
/* 2 */
{
  "_id" : 40,
  "count" : 18
}

```

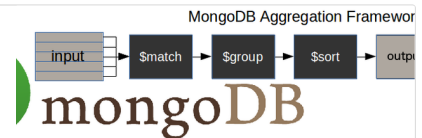
▼ \$lookup stage

Source:

MongoDB Aggregation: tutorial with examples and exercises | Studio 3T

Master the MongoDB aggregation pipeline. Follow along with query examples using the most important aggregation stages, and test your skills!

<https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/#mongodb-lookup>



\$lookup অপারেটর ব্যবহার করা হয় কোন একটি ডকুমেন্টের ভিতরে রেফারেন্স হিসাবে যে আইডি বা ফরেন কি থাকে তার ওপর ভিত্তি করে সেই রেফারেন্স ডাটা কে কারেন্ট ডাটার সাথে মার্জ করা.

যেমন একটি অর্ডার কালেকশনে অর্ডারের মধ্যে ইউজার আইডি রেফারেন্স হিসেবে ব্যবহার করা হয় এখন সেই আইডিকে রেফারেন্স করে ইউজার কে নিয়ে এসে অর্ডার ডকুমেন্টের সাথে ইউজারের ইনফরমেশন মার্জ করে প্রোভাইড করা হয়

from: "data", ফ্রম হচ্ছে যে কালেকশন থেকে ডাটা রেফার করতেছে সেই কালেকশনের নাম

localField: "userId", লোকাল ফিল্ড হচ্ছে কারেন্ট ডকুমেন্ট এর মধ্যে রেফারেন্স ভ্যালুকে কি নামে রাখা হয়েছে

foreignField: "_id", ফরেন ফিল্ড হচ্ছে রেফারেন্স কালেকশনের মধ্যে রেফারেন্স আইডিকে কি নামে দেওয়া আছে

as: "userInfo" এজ হচ্ছে ডকুমেন্ট কে কি হিসেবে নামকরণ করতে চাই.

Example:

```

// Input
db.getCollection("practice-orders").aggregate([
  {
    $lookup: {
      from: "data",
      localField: "userId",
      foreignField: "_id",

```

```

        as: "userInfo"
      }
    }
  })
}

```

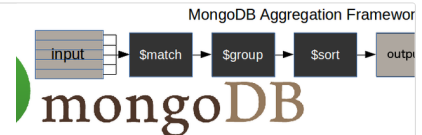
▼ \$facet

Sources:

MongoDB Aggregation: tutorial with examples and exercises | Studio 3T

Master the MongoDB aggregation pipeline. Follow along with query examples using the most important aggregation stages, and test your skills!

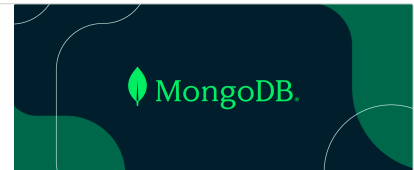
<https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/#mongodb-facet>



\$facet (aggregation)

.leafygreen-ui-zzkys8{font-size:16px;line-height:28px;font-family:'Euclid Circular A','Helvetica Neue',Helvetica,Arial,sans-serif;display:-webkit-inline-box;display:-webkit-inline-flex;display:-ms-inline-flexbox;display:inline-flex;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-

<https://www.mongodb.com/docs/manual/reference/operator/aggregation/facet/>



কোন একটা কালেকশনের উপর রিপোর্ট জেনারেট করার জন্য মাল্টিপল কন্ডিশন ভিত্তিক অপারেশন চালাতে হয় এর জন্য **\$facet** অপারেটর মাল্টিপল Pipeline প্রোভাইড করে প্রতিটি পাইপ লাইনে একাধিক Stage থাকতে পারে যেমন:

```

// Input
db.data.aggregate([
  {
    $facet: {
      // pipeline-1
      "friendsCounts": [
        // stage-1
        { $unwind: "$friends" },
        // stage-2
        { $group: { _id: "$friends", count: { $sum: 1 } } }
      ],
      // pipeline-2
      "educationCount": [
        // stage-1
        { $unwind: "$education" },
        // stage-2
        { $group: { _id: "$education", count: { $sum: 1 } } }
      ],
      // pipeline-3
      "skillsCount": [
        // stage-1
        { $unwind: "$skills" },
        // stage-2
        { $group: { _id: "$skills", count: { $sum: 1 } } }
      ]
    }
  }
])

```

▼ Index, execution time

ইন্ডেক্সিং দুইভাবে করা যায় একটা হচ্ছে মঙ্গলডিবি কম্পাস ব্যবহার করে আরেকটা হচ্ছে কোডের মাধ্যমে.

Indexing করার সময় value গুলোকে ascending অথবা descending অর্ডারে নেওয়ার আগে চিন্তা করতে হবে যে বেলুর উপর ফিল্টারিং করে ইনডেক্সিং করতে চাচ্ছি সেটি অ্যালফাবেটিক্যালি গুরুত্ব দিকে আছে নাকি শেষের দিকে তার উপরে ডিপেন্ড করে ascending অথবা descending অর্ডারে ইনডেক্সিং করলে সময় কম লাগে.

Single index

সিঙ্গেল ফিল্ডের উপর index চালালে সেটাকে বলা হয় single index.

Create new single index

```
// Create index for email field.  
db.getCollection("messive-data").createIndex({email: 1}) // 1 mean Ascending order
```

Drop index

```
db.getCollection("messive-data").dropIndex({email: 1})
```

Multiple / Compound index

সিঙ্গেল ফিল্ডের উপর index চালালে সেটাকে বলা হয় multiple index.

Create multiple index

Go to MongoDB compass, select collection and create multiple index.

or

Create compound index

```
db.getCollection("messive-data").createIndex(  
  { "about": 1, "email": 1 }, //MongoDB 4.2 wildcard index on a specific field { "field.$*" : 1 }, MongoDB 7.0 Compound Wildcard In  
)
```

Drop compound index

```
db.getCollection("messive-data").dropIndex(  
  { "about": 1, "email": 1 }, //MongoDB 4.2 wildcard index on a specific field { "field.$*" : 1 }, MongoDB 7.0 Compound Wildcard In  
)
```

কোন একটি অপারেশন কমপ্লিট করতে কত সময় লাগছে তা বের করার জন্য **.explain("executionStats")** ব্যবহার করলে সম্পূর্ণ অপারেশন ডিটেইলস পাওয়া যায়.

Example:

```
db.getCollection("messive-data").find({ _id: ObjectId("654dbfa0ac03b4ef85f2a90a") })  
.explain("executionStats")
```

▼ Search with indexing

First create index the search field / fields, and type must be use **“text”**

```
// For single field
db.getCollection("messive-data").createIndex({ about: "text"})

// For multiple fields
db.getCollection("messive-data").createIndex({ about: "text", address: "text" })
```

Search

```
db.getCollection("messive-data").find({ $text: { $search: "voluptate" } })
```

\$text means text index.