WELCOME

Building analytics dashboard with MongoDB aggregation pipelines

Aggregation Operations

Aggregation operations process multiple documents and return computed results. You can use aggregation operations to:

- Group values from multiple documents together.
- Perform operations on the grouped data to return a single result.
- Analyze data changes over time.

Aggregation Pipelines

An aggregation pipeline consists of one or more stages that process documents:

- Each stage performs an operation on the input documents. For example, a stage can filter documents, group documents, and calculate values.
- The documents that are output from a stage are passed to the next stage.
- An aggregation pipeline can return results for groups of documents. For example, return the total, average, maximum, and minimum values.

Common Stages

$match - Filter (reduce) the number of documents that is passed to the next stage

$group - Group documents by a distinct key, the key can also be a compound key

$project - Pass documents with specific fields or newly computed fields to the next stage

$sort - Returns the input documents in sorted order

$limit - Limit the number of documents for the next stage

$unwind - Splits an array into into one document for each element in the array

$out - As the last stage, creates/replaces an unsharded collection with the input documents

Common Operators

Operators that return a value based on document data.

Arithmetic Operators
- $abs
- $add
- $multiply
- $subtract
- $trunc

Group Operators
- $sum
- $avg
- $max
- $min
- $first
- $last

Date Operators
- $year
- $month
- $week
- $hour
- $minute
- $second

Operators that return **true** or **false** based on document data.

Comparison Operators
- $eq
- $gt
- $lt
- $gte
- $lte

Boolean Operators
- $and
- $or

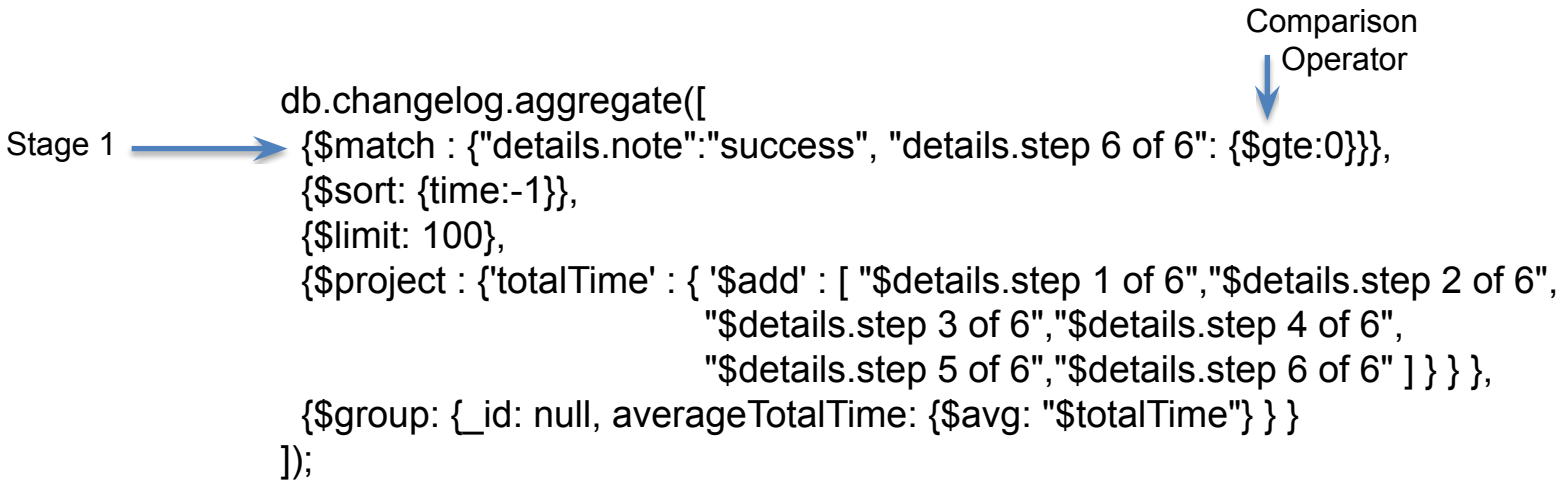
Aggregate()

Purpose: Return the average number of milliseconds to move a chunk for the last one hundred moves.


```
db.changelog.aggregate([
  {$match : {"details.note":"success", "details.step 6 of 6": {$gte:0}}},
  {$sort: {time:-1}},
  {$limit: 100},
  {$project : {'totalTime' : { '$add' : [ "$details.step 1 of 6","$details.step 2 of 6",
                                          "$details.step 3 of 6","$details.step 4 of 6",
                                          "$details.step 5 of 6","$details.step 6 of 6" ] } } },
  {$group: {_id: null, averageTotalTime: {$avg: "$totalTime"} } }
]);
```

$match

Purpose: In the first stage filter only the chunks that moved successfully.



$sort

Purpose: Sort descending so we are prioritizing the most recent moved chunks.

Stage 2 

```
db.changelog.aggregate([
  {$match : {"details.note":"success", "details.step 6 of 6": {$gte:0}}},
  {$sort: {time:-1}},
  {$limit: 100},
  {$project : {'totalTime' : { '$add' : [ "$details.step 1 of 6","$details.step 2 of 6",
                                          "$details.step 3 of 6","$details.step 4 of 6",
                                          "$details.step 5 of 6","$details.step 6 of 6" ] } } },
  {$group: {_id: null, averageTotalTime: {$avg: "$totalTime"} } }
]);
```

$limit

Purpose: Further reduce the number of moves being analyzed because time to move a chunk varies by chunk and collection.

Stage 3 

```
db.changelog.aggregate([
  {$match : {"details.note":"success", "details.step 6 of 6": {$gte:0}}},
  {$sort: {time:-1}},
  {$limit: 100},
  {$project : {'totalTime' : { '$add' : [ "$details.step 1 of 6","$details.step 2 of 6",
                                          "$details.step 3 of 6","$details.step 4 of 6",
                                          "$details.step 5 of 6","$details.step 6 of 6" ] } } },
  {$group: {_id: null, averageTotalTime: {$avg: "$totalTime"} } }
]);
```

$project

Purpose: For each moveChunk document project the sum of the steps to the next stage.

Stage 4 

```
db.changelog.aggregate([
  {$match : {"details.note":"success", "details.step 6 of 6": {$gte:0}}},
  {$sort: {time:-1}},
  {$limit: 100},
  {$project : {'totalTime' : { '$add' : [ "$details.step 1 of 6","$details.step 2 of 6",
                                          "$details.step 3 of 6","$details.step 4 of 6",
                                          "$details.step 5 of 6","$details.step 6 of 6" ] } } },
  {$group: {_id: null, averageTotalTime: {$avg: "$totalTime"} } }
]);
```

$group

Purpose: Return the average number of milliseconds to move a chunk for the last one hundred moves.

```
db.changelog.aggregate([
  {$match : {"details.note":"success", "details.step 6 of 6": {$gte:0}}},
  {$sort: {time:-1}},
  {$limit: 100},
  {$project : {'totalTime' : { '$add' : [ "$details.step 1 of 6","$details.step 2 of 6",
                                          "$details.step 3 of 6","$details.step 4 of 6",
                                          "$details.step 5 of 6","$details.step 6 of 6" ] } } },
  {$group: {_id: null, averageTotalTime: {$avg: "$totalTime"} } }
]);
```

Stage 5 

Requirement -> https://bit.ly/mug-challenge

1. Add this connection string to mongodb compass mongodb+srv://mugjkt:mugjkt@cluster0.oeh2p2u.mongodb.net/

2. Download dashboard html files

Challenge !!!

Tampilkan total kandidat caleg tiap partai

Level : **Easy**

Collection : dapil_1_jakarta

Chart on Dashboard

Tampilkan total kandidat caleg tiap agama

Level : **Easy**

Collection : dapil_1_jakarta

Chart on Dashboard

Top 5 jumlah DPT berdasarkan provinsi

Level : **Easy**

Collection : dpt_1

Tampilkan jumlah caleg perempuan dan laki-laki per partai

Level : **Medium**

Collection : dapil_1_jakarta

Chart on Dashboard

berapa jumlah caleg yang pendidikan terakhirnya SMA

Level : **Medium**

Collection : dapil_1_jakarta

Tampilkan jumlah caleg per dapil/partai

Level : **Medium**

Collection : caleg_2019_2024

Chart on Dashboard

Berapa rata-rata usia caleg per partai ?

Level : **Hard**

Collection : dapil_1_jakarta

Chart on Dashboard

Berapa banyak caleg PKS yang mempunyai pengalaman kerja lebih dari 20 tahun ?

Level : **Hard**

Collection : dapil_1_jakarta

Operator : match, unwind, group, min, max, toInt, subtract, count/sum

Thank you for your time.