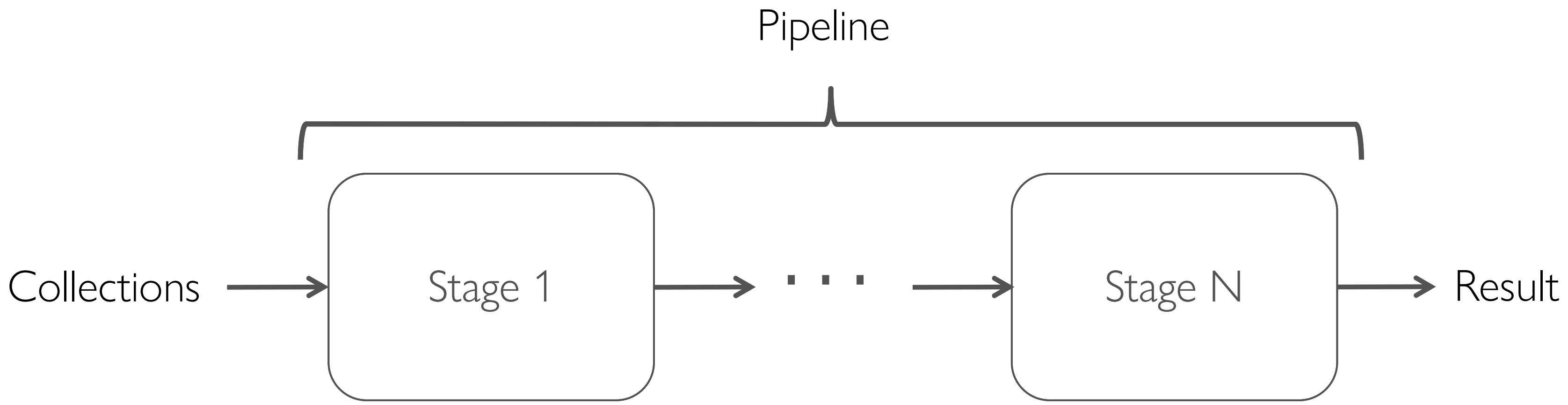
AGGREGATED QUERIES: Stages

- Returns a list of documents resulting from the operations performed at each stage



```
db.collection.aggregate ( [ { <stage> }, ... ] )
```

AGGREGATE QUERIES: $match

- Filters based on specified condition. Use early in the pipeline to improve query performance.
- The syntax is the same as that used for find and findOne . The same operators can be used.

- SYNTAX:

```
{$match: {< query >}}
```

- Example :

```
db.students.aggregate ([
        { $match: {"name.first ": "Perico",
                  " name.second ": "Garcia"}
        }])
```

AGGREGATE QUERIES: $lookup

- Performs a query that combines records from two or more collections (JOIN).

- SYNTAX:

```
{$ lookup :
        { from : < collection to join >,
                        localField : < field from the input documents >,
                          foreignField : < field from the documents of the " from "
collection >,
                        as: <output array field > }
```

- Example :

```
db.university.aggregate ([
        { $lookup : { from : " students " ,
              localField : " name ",
             foreignField : " university ",
           as : " students "}
        }])
```

AGGREGATE QUERIES: $group

- Groups documents based on a specified expression.

- SYNTAX:

```
{ $group: { _id: <expression>,
        <field1>: { <accumulator1>: <expression1> }, ... }
```

- Example :

```
db.students.aggregate ([
        { $match: { branch : " Health "}}
        { $group : {_id: "$branch "
            students_num : {$sum: 1}
        }])
```

AGGREGATE QUERIES: Expression operators

- Accumulators
  - $sum: sum of fields of grouped documents or numbers
  - $avg : average of variables of the grouped documents or numbers
  - $first : returns the value of a field from the first grouped document
  - $last - Returns the value of a field from the last grouped document
  - $max - Returns the maximum value of a field from the grouped documents.
  - $min: Returns the minimum value of a field from the grouped documents.
  - $push : returns a list with the specified field from each of the grouped documents
  - $addToSet : returns a set with the specified field from each of the grouped documents
- Example :

```
db.students.aggregate ([
        { $ group : { _id: "$branch ",
                        students_num : {$sum: 1},
                        total_avg_grade : {$avg : “$avg_score ”},
                       degree : {$addToSet :"$degree "}}
        }])
```

AGGREGATE QUERIES: $unwind

- Breaks down a list of one or more documents, creating as many documents as there are elements in the list.

- SYNTAX:

```
{ $unwind: <field path> }
```

- Example :

```
db.students.aggregate ([
                    { $match: {" name.first ": "Perico"}}
                    { $unwind : "$subjects " } ])
```

AGGREGATE QUERIES: $sort

- Sort the documents.

- SYNTAX:

```
{ $sort: { <field1>: <sort order>, ... } }
```

- Example :

```
db.students.aggregate ([
        { $sort : { avg_grade : 1, fails : -1}}
                        ])
```

AGGREGATE QUERIES: $limit

- Limit the number of documents

- SYNTAX:

```
{ $limit: <positive integer> }
```

- Example :

```
db.students.aggregate ([
        { $ limit : 15 } ])
```

ADDED QUERIES: $skip

- Skips the first few documents and returns the rest.

- SYNTAX:

```
{ $skip: <positive integer> }
```

- Example :

```
db.students.aggregate ([
        { $ skip : 5 } ])
```

ADDED QUERIES: $geoNear

- Sorts documents from closest to farthest from a specified point

- SYNTAX:

```
{ $geoNear : { < geoNear options > } }
```

- Example :

```
db.students.aggregate ([{$geoNear : {
        near: { type: "Point",
                      coordinates: [ -73.99279 , 40.719296 ] },
                       maxDistance : 9,
        query: { branch: "health" },
        distanceField : "dist.calculated ",
        includeLocs : "dist.location ",
        num: 5,
        spherical: true
        }
   }
])
```

AGGREGATE QUERIES: $out

- Writes the result of the aggregate query to a collection. It must be last in the pipeline.
- If a collection with that name does not exist, it creates it, otherwise it overwrites it.

- SYNTAX:

```
{ $out: "<output-collection>" }
```

- Example :

```
db.students.aggregate ([
                { $match: {" branch ": " Health "}}
                { $out : " health_students " } ])
```

ADDED QUERIES: $project

- Remodel documents (adding or removing fields). For each document that comes in, one goes out.
- SYNTAX:

```
{$project : { <specifications> }}
              < field >: 1/0 or true/false
              < field >: < expression >
```

  - The _id field is added by default, the rest of the fields must be specified.
  - Operators can be used to create new fields.
- Example :

```
db.students.aggregate ([
    { $project :{"name.first":1,
                        branch:1,
                 abrev_subjects : {$substr :["$subjects",0,3]}} }])
```

AGGREGATE QUERIES: Expression operators

- Boolean operators (true or false):
  - $and: and
  - $or : or
  - $not : no
- Comparison operators (true or false) :
  - $cmp : returns 0 if they are equal, 1 if the first is greater than the 2nd and -1 if the first value is less than the second
  - $eq : is equal
  - $gt : greater than
  - $gte : greater than or equal to
  - $in : in the list
  - $lt : less than
  - $lte : less than or equal to
  - $ne : different from

AGGREGATE QUERIES: Expression operators

- Operators on sets (do not work with nested arrays):
  - $setEquals : returns true if they are equal, false otherwise
  - $setIntersection : returns the common elements of two or more arrays
  - $setUnion : returns the union of two or more arrays
  - $setDifference : returns the elements that appear in the first array but not in the second
  - $setIsSubset : returns true if the first array is a subset of the second array
  - $anyElementTrue : returns true if any element in the array is true
  - $allElementsTrue : : returns true no element in the array is false

AGGREGATE QUERIES: Expression operators

- Arithmetic operators:
  - $add : Adds two fields/numbers or a variable/number to a date.
  - $subtract : Subtracts two fields/numbers or a variable/number from a date.
  - $multiply : multiply two fields/numbers
  - $divide: divide two fields/numbers
  - $mod : returns the remainder when dividing two fields/numbers

AGGREGATE QUERIES: Expression operators

- Operators on strings:
  - $concat : Concatenates two or more fields/strings.
  - $substr - Returns a substring of the specified field/string.
  - $toLower : Changes a field/string to lowercase
  - $toUpper: Changes a field/string to uppercase
  - $strcasecmp : Performs a case-insensitive comparison between two strings. Returns zero if the strings are equal, 1 if the first string is greater than the second, and -1 otherwise.

AGGREGATE QUERIES: Expression operators

- Operators on lists:
  - $size : Returns the size of a list.
- Operators on variables
  - $map : Applies an expression to each of the elements in a list and returns a list with the results.
  - $let : allows you to apply expressions on dynamically created variables.
- Operators on literals
  - $literal: Returns a value without parsing it. Used when you don't want a string to be evaluated that could be mistaken for an expression.

AGGREGATE QUERIES: Expression operators

- Operators on dates:
  - $dayOfYear : 1-366
  - $dayOfMonth : 1-31
  - $dayOfWeek : 1(Sunday)-7
  - $year
  - $month : 1-12
  - $week : 0-53
  - $hour : 0-23
  - $minute: 0-59
  - $second : 0-59
  - $millisecond : 0-999

AGGREGATE QUERIES: Expression operators

- Conditional expressions:
  - $cond : Performs an " if-then " with a boolean expression. If the expression is true, the first specified value is returned; otherwise, the opposite is true.
  - $ifnull : evaluates expression and if it is null , returns a specified value