$match:

* can act as a filter, to only pass through documents matching certain criteria.
* Example:

stage\_match\_title = {

"$match": {

"title": "A Star Is Born"

}

}

$limit:

* can Limit the Number of Results.
* Example:

stage\_limit\_1 = { "$limit": 1 }

$[lookup](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/):

* like a JOIN in a relational database
* Example:

stage\_lookup\_comments = {

"$lookup": {

"from": "comments",

"localField": "\_id",

"foreignField": "movie\_id",

"as": "related\_comments",

}

}

$addFields, $size:

* Add a field , gives size
* Example:

stage\_add\_comment\_count = {

"$addFields": {

"comment\_count": {

"$size": "$related\_comments"

}

}

}

# Match movie documents with more than 2 comments:

stage\_match\_with\_comments = {

"$match": {

"comment\_count": {

"$gt": 2

}

}

}

$group, $sum:

* stage groups input documents by the specified \_id expression and output one document for each unique \_id value.
* This will add a movie\_count field, containing the result of adding 1 for every document in the group. In other words, it counts the number of movie documents in the group.
* Example:

stage\_group\_year = {

"$group": {

"\_id": "$year",

# Count the number of movies in the group:

"movie\_count": { "$sum": 1 },

}

}

$type:

* only match documents with a numeric year
* Example:

stage\_match\_years = {

"$match": {

"year": {

"$type": "number",

}

}

}

$sort:

stage\_sort\_year\_ascending = {

"$sort": {"\_id": pymongo.ASCENDING}

}

All: [Aggregation Pipeline Stages — MongoDB Manual](https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/?_ga=2.113050636.948779982.1700359039-14261387.1696792465)

Full documentation: [Aggregation Pipeline — MongoDB Manual](https://www.mongodb.com/docs/manual/core/aggregation-pipeline/?_ga=2.116075278.948779982.1700359039-14261387.1696792465)