Then the second `$skip` stage can coalesce into the first `$skip` stage and result in a single `$skip` stage where the skip amount `7` is the sum of the two initial limits `5` and `2`.

```
{ $skip: 7 }
```

`$match` + `$match` Coalescence When a `$match` immediately follows another `$match`, the two stages can coalesce into a single `$match` combining the conditions with an `$and`. For example, a pipeline contains the following sequence:

```
{ $match: { year: 2014 } },
{ $match: { status: "A" } }
```

Then the second `$match` stage can coalesce into the first `$match` stage and result in a single `$match` stage

```
{ $match: { $and: [ { "year" : 2014 }, { "status" : "A" } ] } }
```

Examples

The following examples are some sequences that can take advantage of both sequence reordering and coalescence. Generally, coalescence occurs *after* any sequence reordering optimization.

`$sort` + `$skip` + `$limit` Sequence A pipeline contains a sequence of `$sort` followed by a `$skip` followed by a `$limit`:

```
{ $sort: { age : -1 } },
{ $skip: 10 },
{ $limit: 5 }
```

First, the optimizer performs the *$skip + $limit Sequence Optimization* (page 447) to transforms the sequence to the following:

```
{ $sort: { age : -1 } },
{ $limit: 15 }
{ $skip: 10 }
```

The *$skip + $limit Sequence Optimization* (page 447) increases the `$limit` amount with the reordering. See *$skip + $limit Sequence Optimization* (page 447) for details.

The reordered sequence now has `$sort` immediately preceding the `$limit`, and the pipeline can coalesce the two stages to decrease memory usage during the sort operation. See *$sort + $limit Coalescence* (page 448) for more information.

`$limit` + `$skip` + `$limit` + `$skip` Sequence A pipeline contains a sequence of alternating `$limit` and `$skip` stages:

```
{ $limit: 100 },
{ $skip: 5 },
{ $limit: 10 },
{ $skip: 2 }
```

The *$skip + $limit Sequence Optimization* (page 447) reverses the position of the `{ $skip: 5 }` and `{ $limit: 10 }` stages and increases the limit amount: