Chapter 2: Basic Aggregation - Utility Stages

Lab - Bringing it all together

Back to the Question

One possible solution is below.

COPY

```
db.movies.aggregate([
  {
    $match: {
      year: { $gte: 1990 },
      languages: { $in: ["English"] },
      "imdb.votes": { $gte: 1 },
      "imdb.rating": { $gte: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      title: 1,
      "imdb.rating": 1,
      "imdb.votes": 1,
      normalized_rating: {
        $avg: [
          "$imdb.rating",
          {
            $add: [
              1,
              {
                $multiply: [
                  9,
                  {
                    $divide: [
                      { $subtract: ["$imdb.votes", 5] },
                      { $subtract: [1521105, 5] }
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }
    }
  },
  { $sort: { normalized_rating: 1 } },
```

```
  { $limit: 1 }
])
```

We start by applying the `$match` filtering:

COPY

```
{
  $match: {
    year: { $gte: 1990 },
    languages: { $in: ["English"] },
    "imdb.votes": { $gte: 1 },
    "imdb.rating": { $gte: 1 }
  }
}
```

And within the `$project` stage we apply the scaling and normalizating calculations:

COPY

```
{
  $project: {
    _id: 0,
    title: 1,
    "imdb.rating": 1,
    "imdb.votes": 1,
    normalized_rating: {
      $avg: [
        "$imdb.rating",
        {
          $add: [
            1,
            {
              $multiply: [
                9,
                {
                  $divide: [
                    { $subtract: ["$imdb.votes", 5] },
                    { $subtract: [1521105, 5] }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  }
},
```

in a new computed field `normalized_rating`.

The first element of the result, after sorting by `normalized_rating` is **The Christmas Tree**, the expected correct answer.

Proceed to next section