**Performance Measurements:**

The performance measurement in MongoDB is quite different compared to the MySQL performance measurements. The MongoDB Compass helps to visualize the processing time of a query to fetch the documents using the option Explain. It also produces a little more vital information about the performance measurements as a Query Performance Summary that includes number of documents returned, documents examined, execution time, sorted in memory and examined index key. While the shell interface provides a comprehensive information about the performance measurements for the provided query by processing every stages of the aggregation pipeline using the query command .explain(“executionStatistics”).

**Performance Measurement comparisons: MySQL vs MongoDB queries**

1. **Locations with the Highest Number of Remote Work Opportunities**

**Query:**

db.Job\_postings.aggregate([{ $match: { remote\_allowed: 1 } },

{$group: {\_id: "$location", totalRemoteJobs: { $sum: 1 }}},

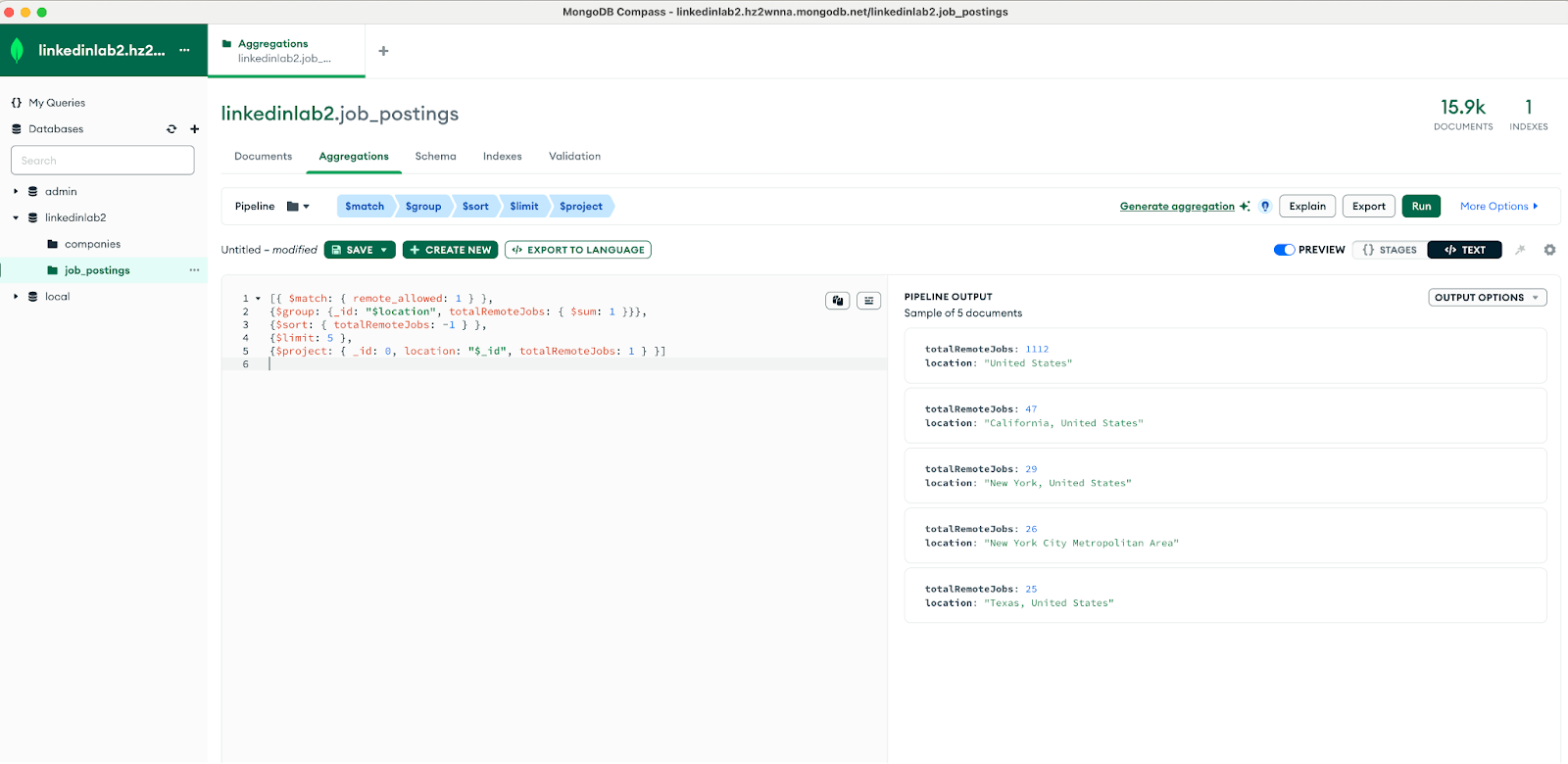
{$sort: { totalRemoteJobs: -1 } },

{$limit: 5 },

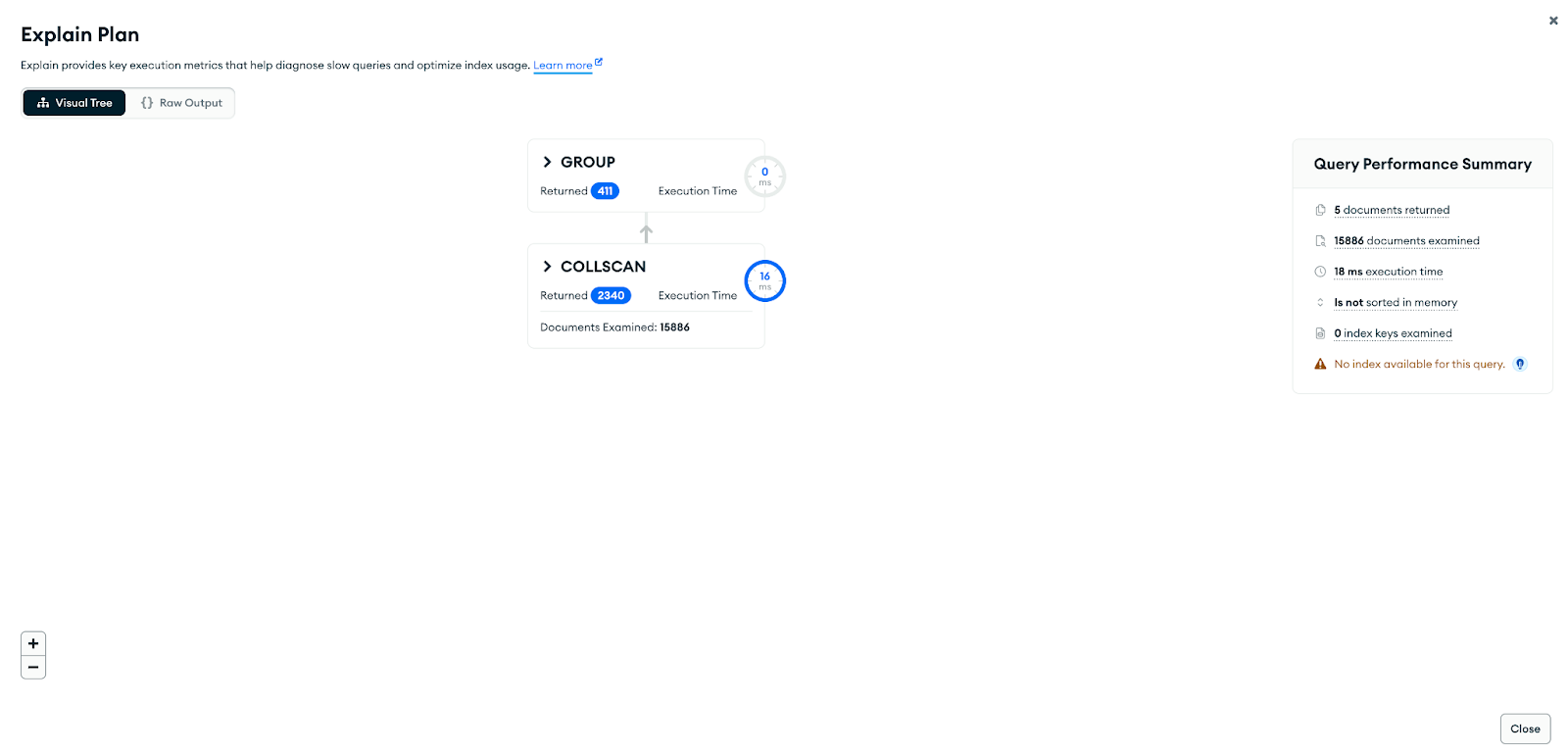
{$project: { \_id: 0, location: "$\_id", totalRemoteJobs: 1 } }])

**Checking performance measurements on MongoDB Compass**

**Creating aggregation pipelines using “</>Text” option:**



**Explain Plan : Visual Tree view**



1. **Top 5 Locations with the Sponsored Job Postings**

**Query:**

db.Job\_postings.aggregate([{ $match: { sponsored: 1 } },

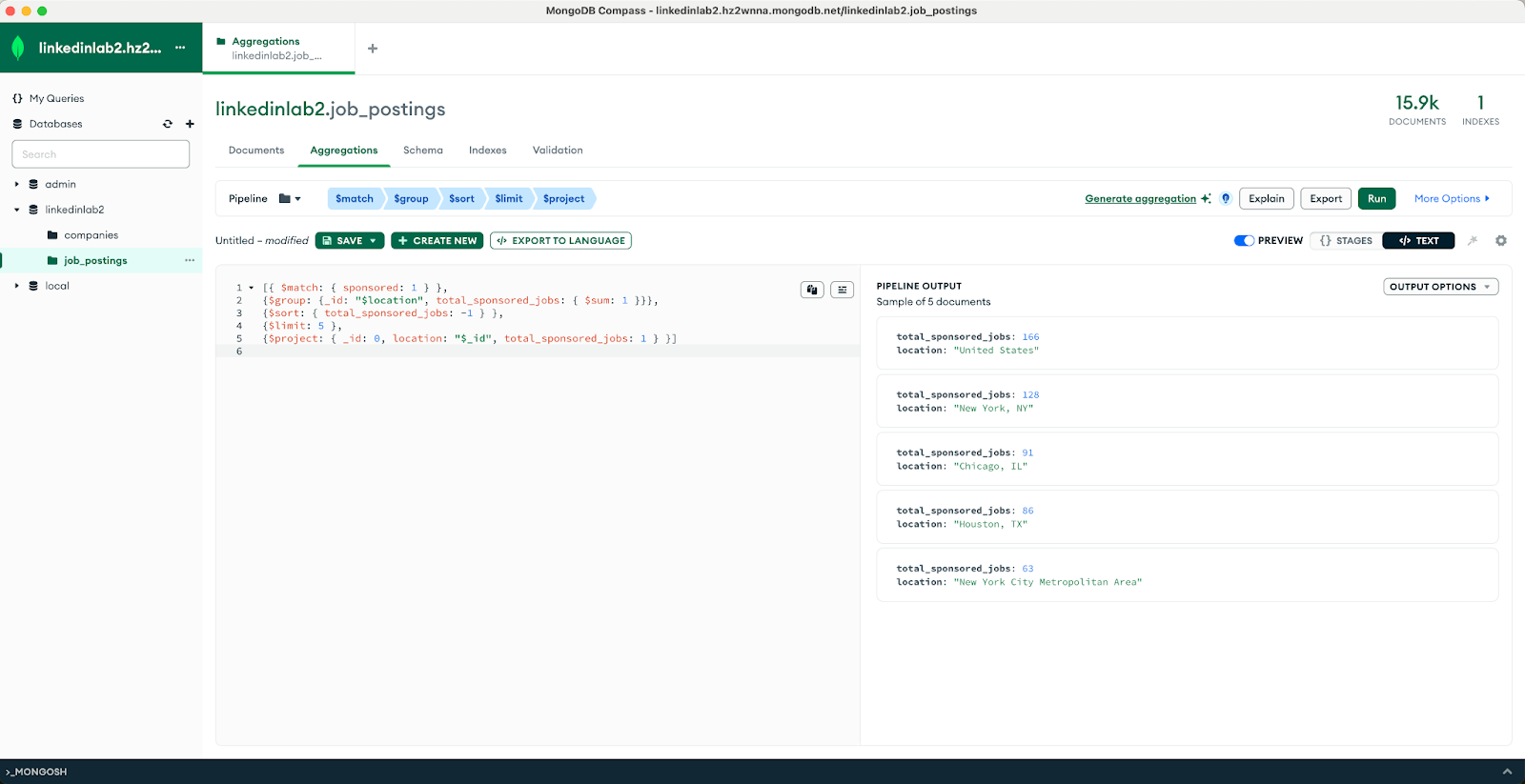
{$group: {\_id: "$location", total\_sponsored\_jobs: { $sum: 1 }}},

{$sort: { total\_sponsored\_jobs: -1 } },

{$limit: 5 },

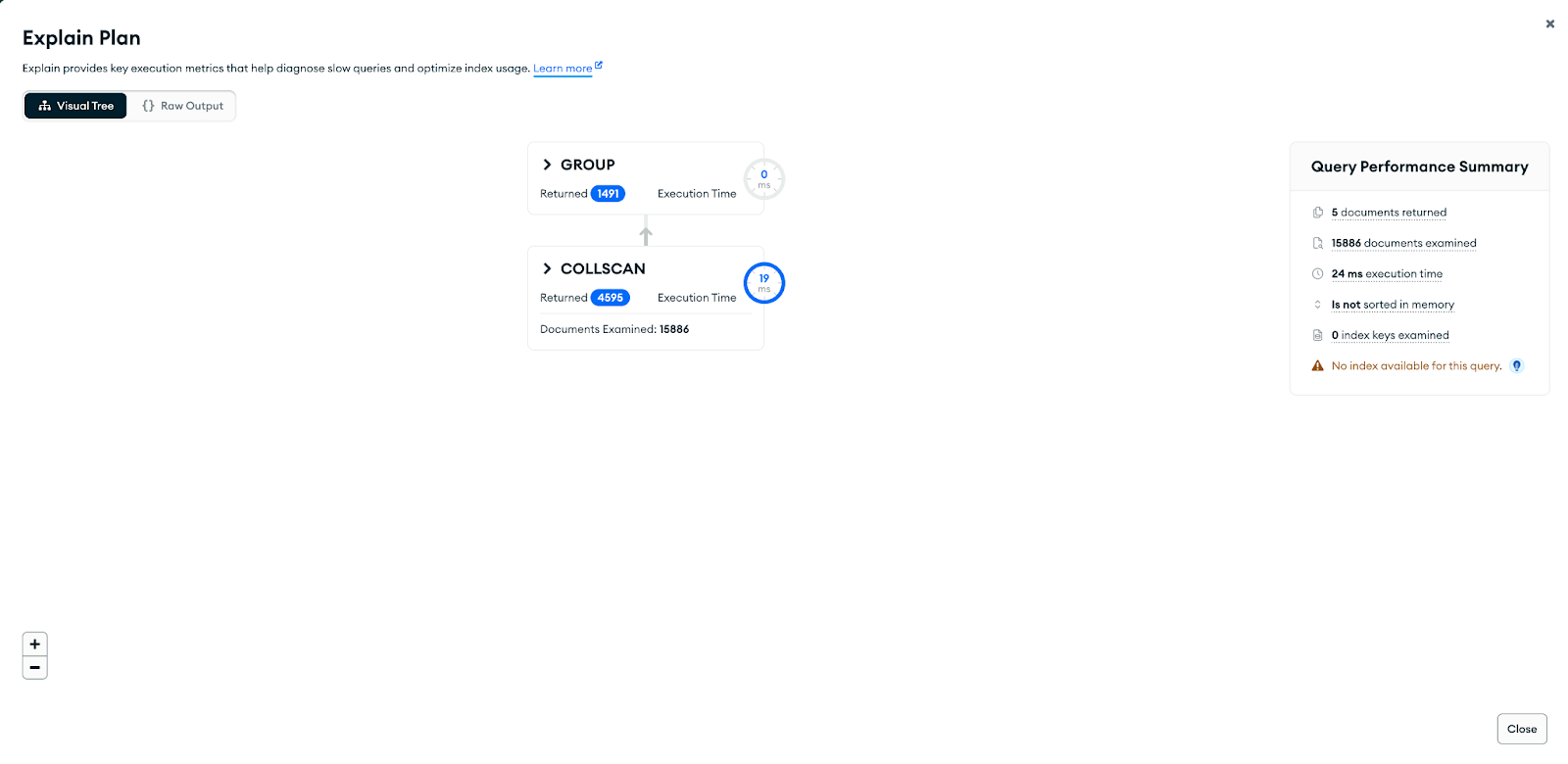
{$project: { \_id: 0, location: "$\_id", total\_sponsored\_jobs: 1 } }])

**Checking performance measurements on MongoDB Compass**



**Displaying 2 different views of the performance measurements:**

1. **Explain Plan : Visual Tree**



**(2) Explain Plan: Raw Output**



1. **Company with highest employee count where specialities is “analytics”**

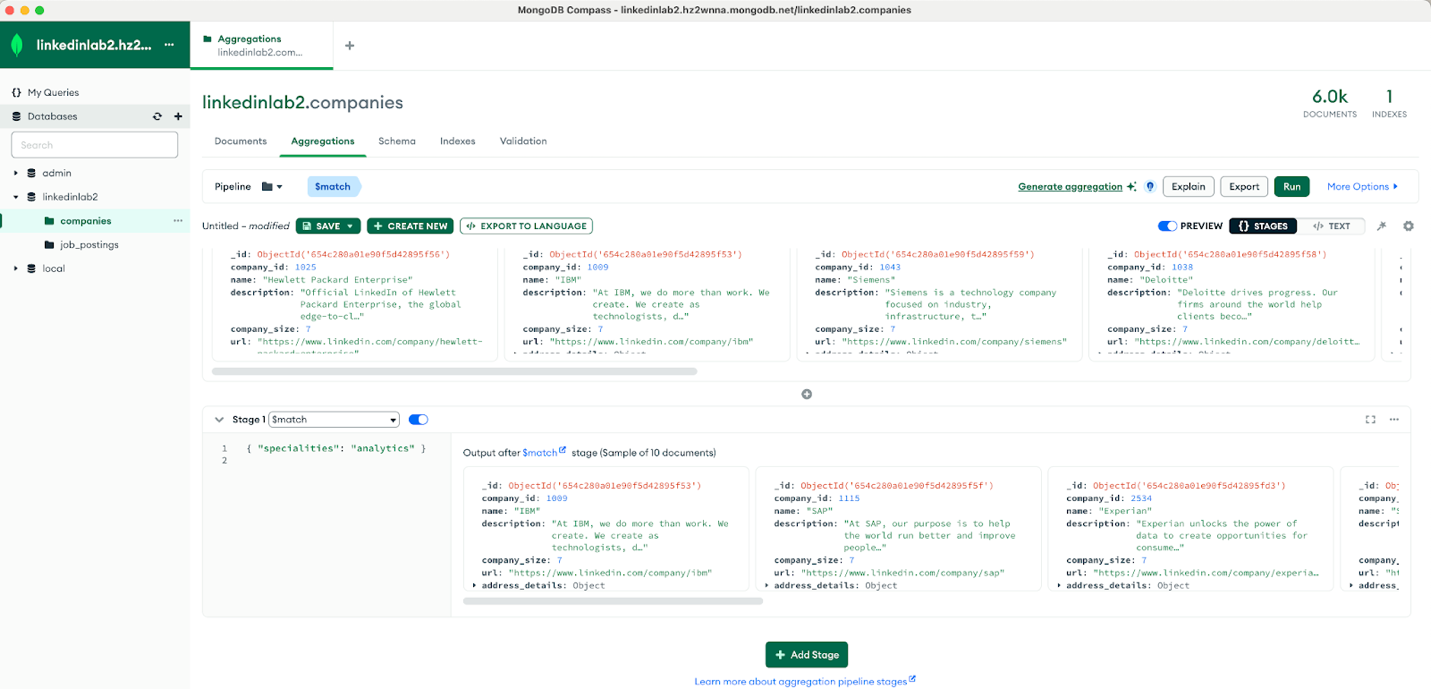
**Query:**

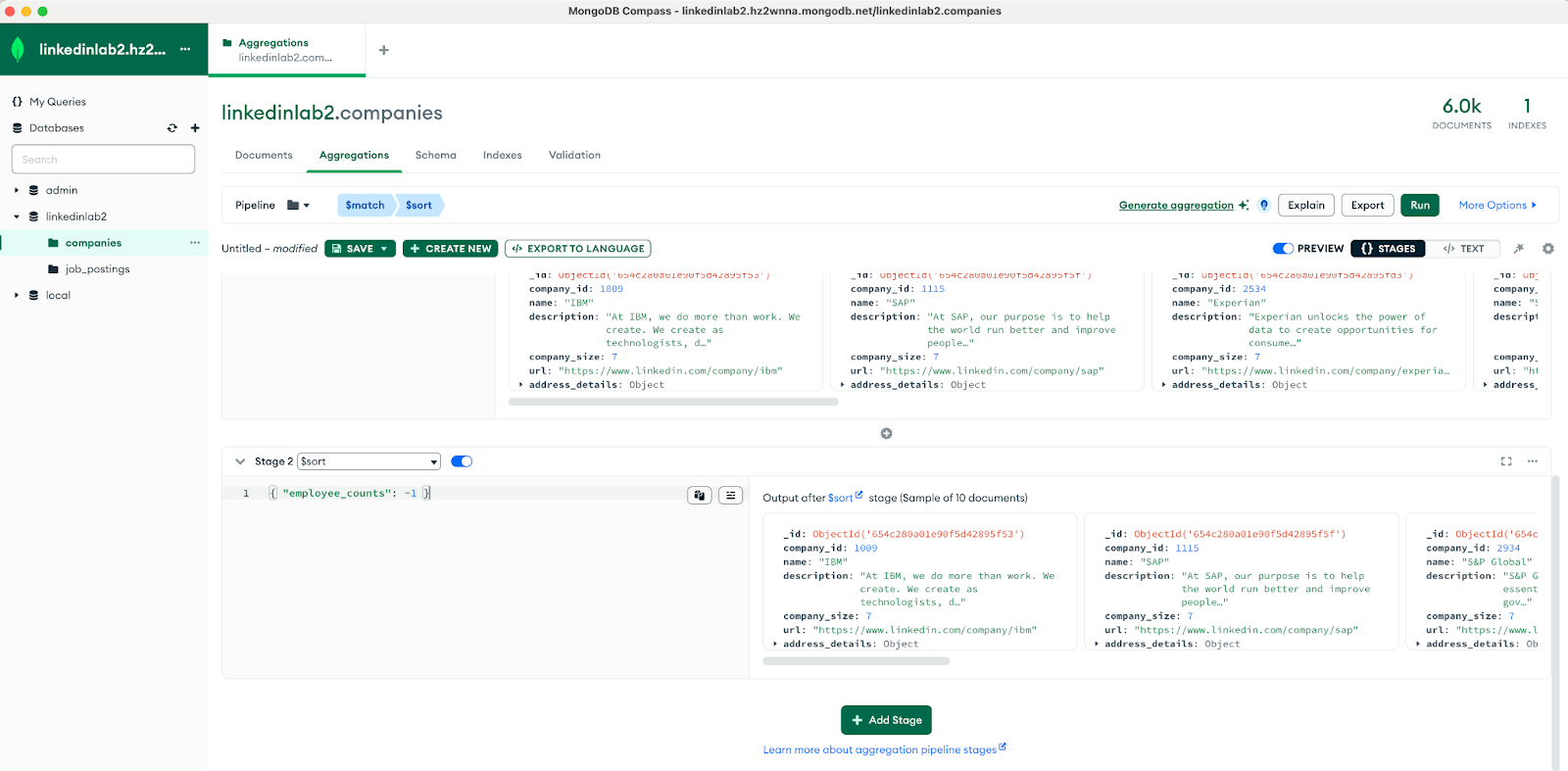
db.Companies.aggregate([{$match: { "specialities": "analytics" } },{$sort: { "employee\_counts": -1 }},{

$limit: 1},{$project: {\_id: 0,company\_name: "$name",company\_size: "$company\_size",employeeCount: "$employee\_counts"}}])

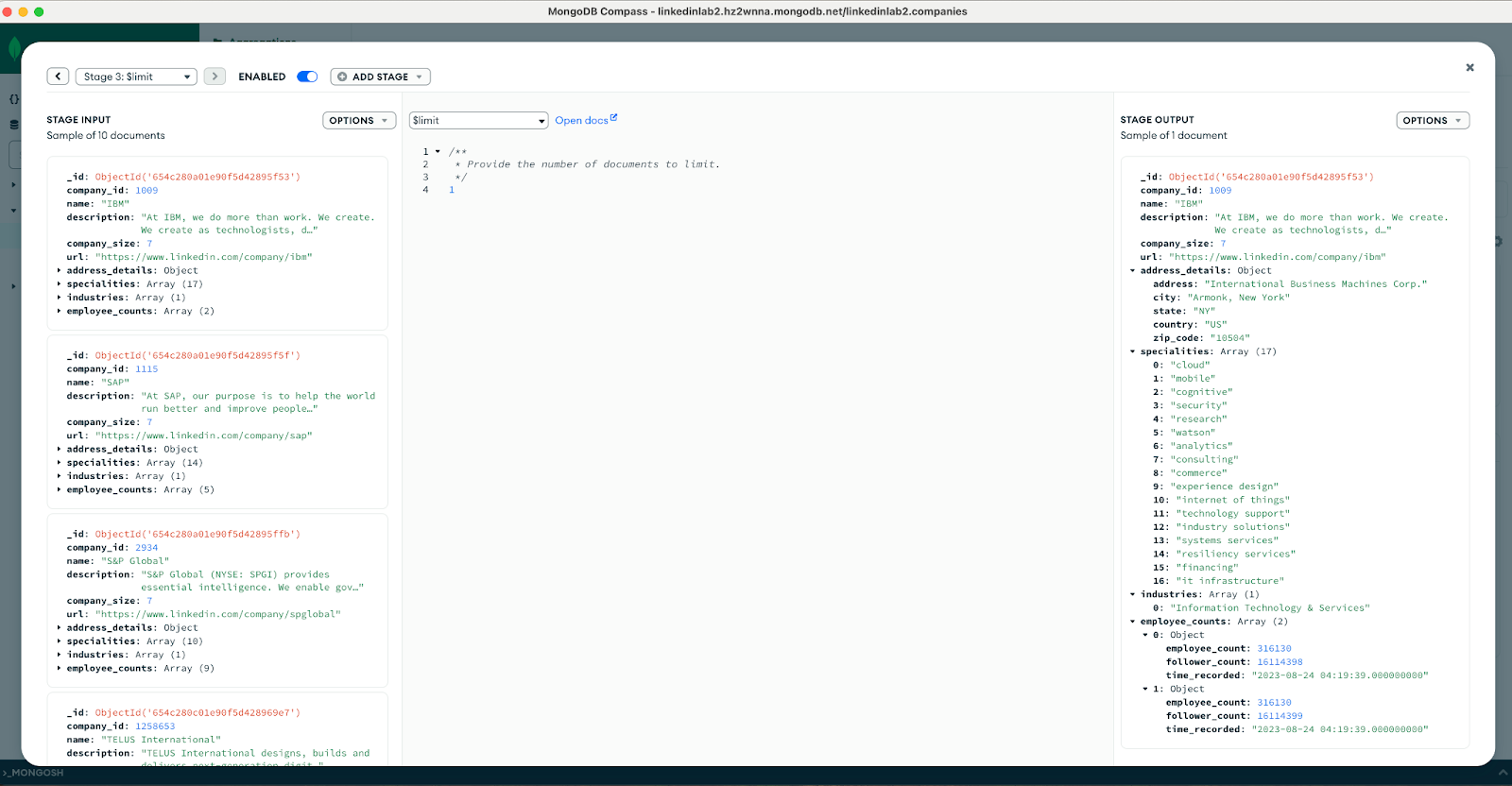
**Creating Pipeline Stages using “{}Stages” option**

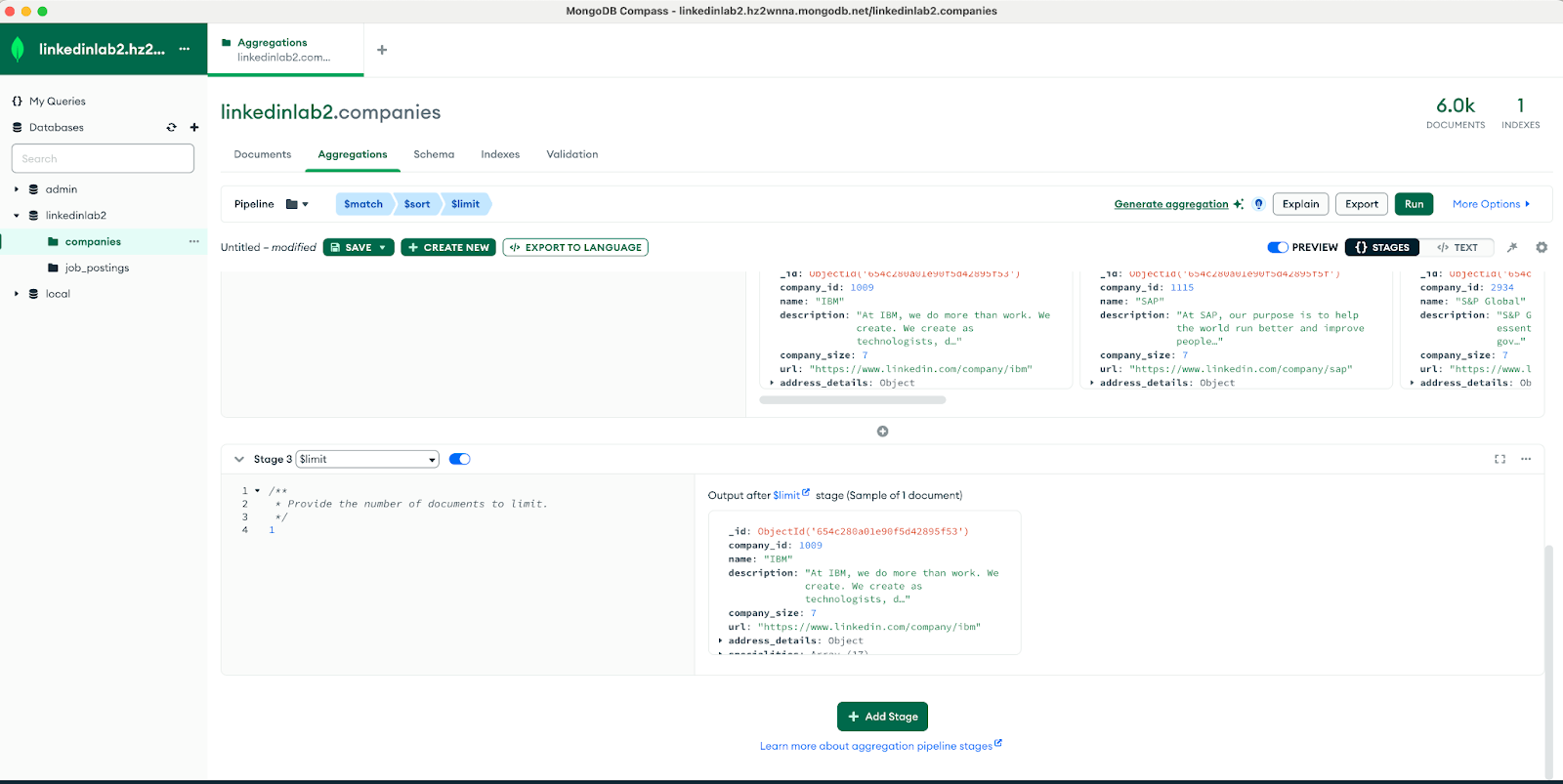
**Stage1: $match**

**Stage2: $sort**

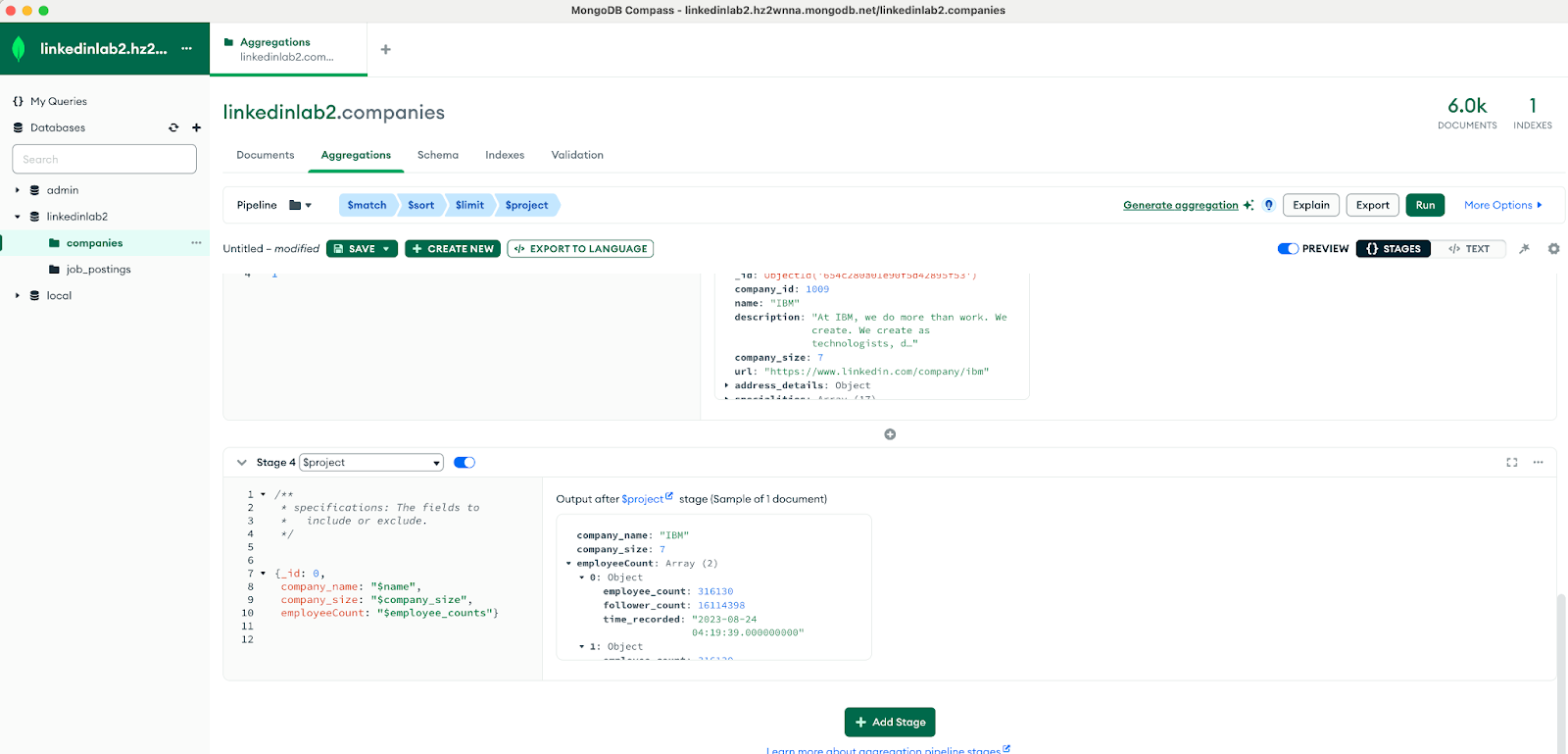


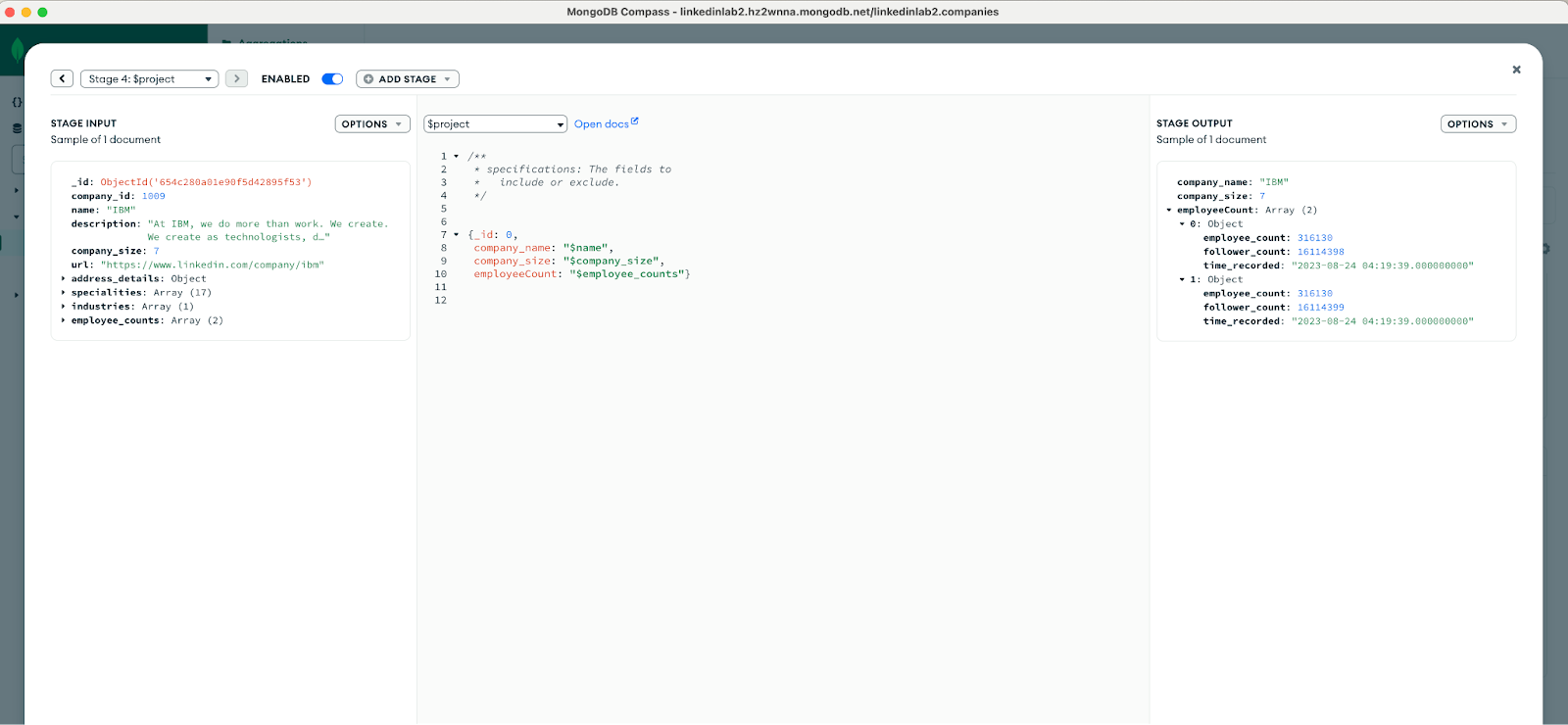
**Stage3: $limit**



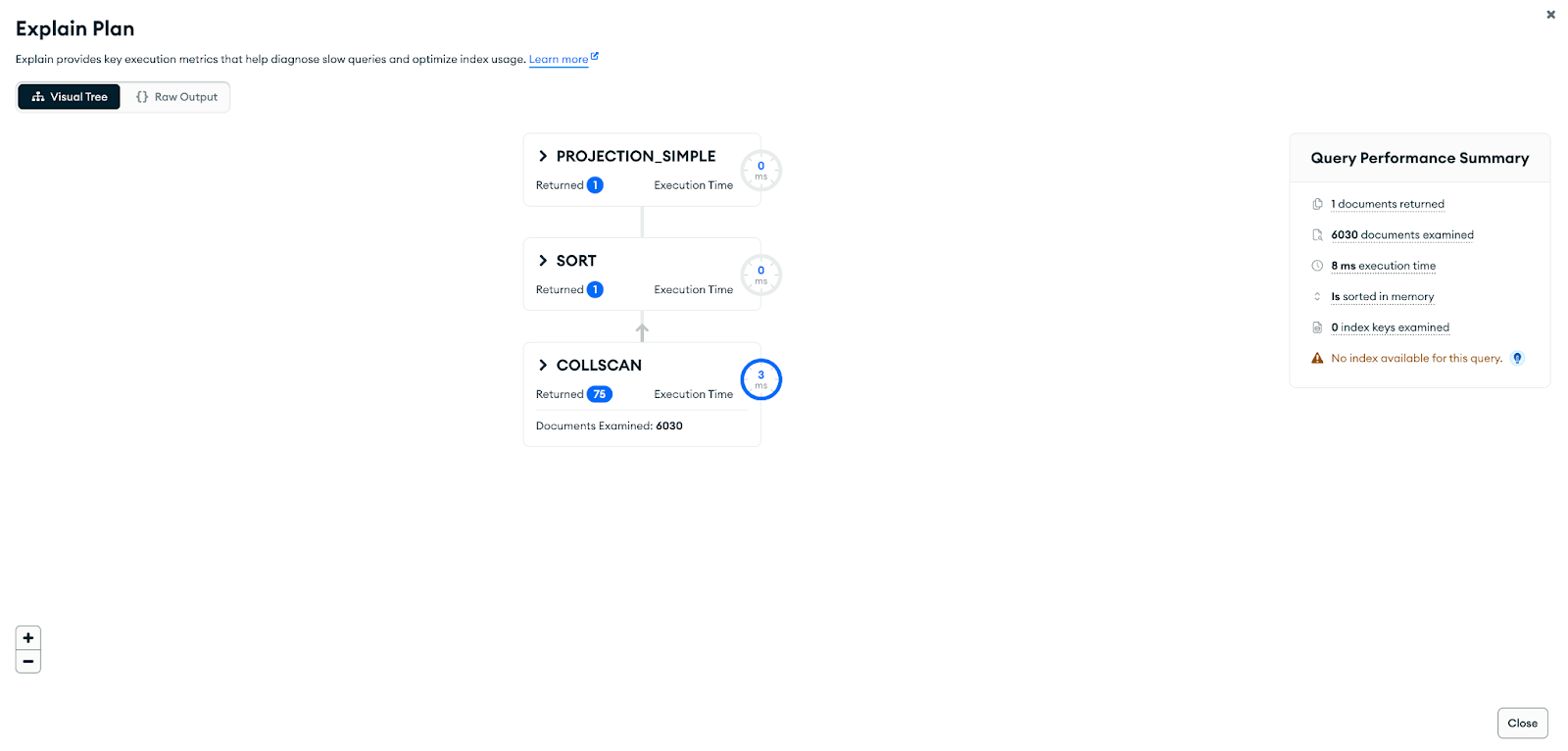


**Stage4: $project**





**Explain Plan : Visual Tree**



**From the above displayed output, below are the performance output scripts from various pipeline stages:**

**PROJECTION\_SIMPLE:**

{

"stage": "PROJECTION\_SIMPLE",

"nReturned": 1,

"executionTimeMillisEstimate": 3,

"works": 6033,

"advanced": 1,

"needTime": 6031,

"needYield": 0,

"saveState": 7,

"restoreState": 7,

"isEOF": 1,

"transformBy": {

"company\_size": 1,

"employee\_counts": 1,

"name": 1,

"\_id": 0

}

}

**SORT:**

{

"stage": "SORT",

"nReturned": 1,

"executionTimeMillisEstimate": 3,

"works": 6033,

"advanced": 1,

"needTime": 6031,

"needYield": 0,

"saveState": 7,

"restoreState": 7,

"isEOF": 1,

"sortPattern": {

"employee\_counts": -1

},

"memLimit": 33554432,

"limitAmount": 1,

"type": "simple",

"totalDataSizeSorted": 0,

"usedDisk": false,

"spills": 0

}

**COLLSCAN:**

{

"stage": "COLLSCAN",

"filter": {

"specialities": {

"$eq": "analytics"

}

},

"nReturned": 75,

"executionTimeMillisEstimate": 3,

"works": 6031,

"advanced": 75,

"needTime": 5955,

"needYield": 0,

"saveState": 7,

"restoreState": 7,

"isEOF": 1,

"direction": "forward",

"docsExamined": 6030

}

**Summary Performance comparision: MySQL vs MongoDB:**  
Comparison of the performance and speed between MySQL and MongoDB queries are purely based on the query’s requirement and the amount of background process to fetch the contents from the tables/collections respectively. It is highly project centric and the time taken to traverse the volume of data to retrieve the necessary information. Here, the Mongodb query performances (using MongoDB Atlas Shell and Compass) were much faster in retrieving the results than the MySQL queries (using MySQL workbench).

Here approximately, MySQL queries showed slower performance over MongoDB queries to fetch data even though the collscan was internally used by the application.

|  |  |  |
| --- | --- | --- |
| **Query Requirement** | **MySQL performance** | **MongoDB performance** |
| Query 1 | 41 ms | 16 ms |
| Query 2 | 41 ms | 19 ms |
| Query 3 | 40 ms | 3 ms |