

Christina DiMaggio

November 22, 2024

## Assignment 5

// 1. Over how many years was the unemployment data collected?

The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists 'myMongoDB' with sub-entries 'Test', 'Amazon', 'Unemployment', 'admin', 'config', 'local', 'moviesdb', and 'social\_networking'. The 'Unemployment' collection is selected. The main panel shows the 'Aggregations' tab with a pipeline of two stages. Stage 1 is a \$group stage with a key of '\$Year' and a count of '\$\_id'. Stage 2 is a \$count stage. The result shows 'Years data was collected: 27'.

```
1 // **
2 * _id: The id of the group.
3 * fieldN: The first field name.
4 */
5 {
6   _id: "$Year",
7   }
8 }
```

```
1 // **
2 * Provide the field name for the count.
3 */
4 'Years data was collected'
```

The unemployment data was collected over 27 years. The aggregation first groups data by the year field. In this stage, all the specific years in the dataset can be found and returned as documents, from 1990 to 2016. Then count is used to retrieve the number of documents in the collection that match the previous query. In this case, 27 years were returned from the group aggregate, so the count aggregate reported 27.

// 2. How many states were reported on in this dataset?

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' panel lists the database 'myMongoDB' and the collection 'Unemployment'. The main panel displays the 'Aggregations' tab for the 'Unemployment' collection. The aggregation pipeline consists of two stages:

- Stage 1: \$group** (highlighted with a red box). The pipeline is defined as:

```
1 /**
2  * _id: The id of the group.
3  * fieldN: The first field name.
4  */
5 {
6   _id: "$State"
7 }
```
- Stage 2: \$count** (highlighted with a red box). The pipeline is defined as:

```
1 /**
2  * Provide the field name for the count.
3  */
4 "State Count"
```

The results panel shows 'State Count : 47'. The bottom status bar indicates the time is 4:40 PM on 11/22/2024.

There are 47 states reported in this dataset. The aggregation first groups data by the state field. In this stage, all the specific states in the dataset can be found and returned as documents. Then count is used to retrieve the number of documents in the collection that match the previous query. In this case, 47 different states were returned from the group aggregate, so the count aggregate reported 47.

// 3. What does this query compute? `db.unemployment.find({Rate : {$lt: 1.0}}).count()`

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' panel shows a connection to 'myMongoDB'. The main panel displays the 'unemployment' collection with 885.5K documents. The 'Aggregations' tab is active, showing a pipeline with two stages. Stage 1 is '\$match' with a query filter `{Rate: {$lt: 1.0}}`. Stage 2 is '\$count' with an output field 'Rates less than 1'. The result shows 657 documents matching the criteria.

There are 657 documents in this dataset that have an unemployment rate of less than 1.0. The aggregation first matches data by the field rate if it is less than 1.0. In this stage, all the specific matches to this query in the dataset can be found and returned as documents. Then count is used to retrieve the number of documents in the collection that match the previous query. In this case, 657 entries of data have an unemployment rate of less than 1.0, so the count aggregate reported is 657.

**`db.unemployment.find({Rate : {$lt: 1.0}}).count()`**

- MongoDB will search the unemployment collection for the documents that match the specific criteria:
  - o Rate less than 1.0 is a query filter. Documents where the Rate field is less than 1.0 are returned
  - o Count tells MongoDB to return the count of the number of documents to match the filter criteria instead of returning the documents themselves.

// 4. Find all counties with unemployment rate higher than 10%

The screenshot shows the MongoDB Compass interface with an aggregation pipeline applied to the 'Unemployment' collection. The pipeline consists of three stages: \$group, \$match, and \$sort. The \$group stage groups data by county and calculates the average unemployment rate. The \$match stage filters for counties where the average rate is greater than 10%. The \$sort stage sorts the results in descending order of the average rate. The output shows a list of counties with their IDs and average rates.

County	avgRate
Imperial County	23.10274914089347
Starf County	20.932150590388372
Presidio County	20.33568281938326
Maverick County	19.669603524229075
Luna County	17.909876543208876
Zavala County	16.93700440528634
Colusa County	16.62233676975945
East Carroll Parish	15.788118811881187
Willacy County	15.736123348017621
Wilcox County	15.100925925925925

There are 105 counties with unemployment rates higher than 10%. The output shows a sample of the documents (image above). The aggregation groups data by the country and its average rate of unemployment over years data was collected. This signifies they have an overall unemployment rate higher than 10%. The aggregation then matches data by the field rate if it is greater than 10%. In this stage, all the specific matches to this query in the dataset can be found and returned as documents. In this stage, all the counties that match the filter of the average rate above 10% in the dataset can be found and returned as documents. Sort -1 is used to order the results from the county with the highest unemployment rate to the lowest.

// 5. Calculate the average unemployment rate across all states.

The screenshot shows the MongoDB Compass interface with the 'Unemployment' collection selected. The 'Aggregations' tab is active, displaying a two-stage pipeline. Stage 1 is a '\$group' operation that groups data by state and calculates the average rate. Stage 2 is another '\$group' operation that calculates the overall average rate across all states. The final result is shown in the 'ALL RESULTS' section.

```
1 // **
2 * _id: The id of the group.
3 * fieldN: The first field name.
4 */
5 {
6   _id: "$State",
7   avgRate: {
8     $avg: "$Rate"
9   }
10 }
```

```
1 // **
2 * _id: The id of the group.
3 * fieldN: The first field name.
4 */
5 {
6   _id: "overall",
7   overallAvgRate: {
8     $avg: "$avgRate"
9   }
10 }
```

ALL RESULTS OUTPUT OPTIONS

Showing 1 - 1 count results

VIEW

OverallAvgRate: 6.232931124517331

The average overall unemployment rate across all states in the data set is 6.23%. The aggregation first groups data by the state field and creates the average rate for each state in the dataset. In this stage, all the specific states and their respective average rates from the dataset can be found and returned as documents. These documents are then averaged together to create the total overall unemployment rate found across all states from the data set, 6.23%.

// 6. Find all counties with an unemployment rate between 5% and 8%.

The screenshot shows the MongoDB Compass interface with an aggregation pipeline applied to the 'Unemployment' collection. The pipeline consists of two stages: Stage 1 (\$group) and Stage 2 (\$match). The output shows a list of documents with their IDs and average unemployment rates.

**Stage 1 (\$group):**

```
1 /**
2  * _id: The id of the group.
3  * fieldN: The first field name.
4  */
5 {
6   _id: "$County",
7   avgRate: {
8     $avg: "$Rate"
9   }
10 }
```

**Stage 2 (\$match):**

```
1 /**
2  * query: The query in MQL.
3  */
4 {
5   avgRate: {
6     $gte: 5,
7     $lte: 8
8   }
9 }
```

**Output Results:**

_id	avgRate
"West Feliciana Parish"	6.282976297629763
"Washington County"	5.938779395296753
"Acadia Parish"	6.437623762376238
"Cross County"	7.899874874874874
"Pecos County"	6.211813215859831
"Scotland County"	7.558641975398642
"Pennington County"	5.853395061728395
"Doomie County"	7.856172839506172
"Neesho County"	5.79328987654321
"Crane County"	5.884845814977973
"Solano County"	6.874226894123712
"Rutherford County"	

There are 945 counties with an unemployment rate between 5% and 8%. The output shows a sample of documents (image above). The aggregation groups data by the country and its average rate of unemployment over years data was collected. The aggregation then matches data by the field rate if it is greater than or equal to 5% or less than or equal to 8%. In this stage, all the specific matches to this query in the dataset can be found and returned as documents. This signifies they have an overall unemployment rate between 5% and 8%. In this stage, all the counties that match the filter in the dataset can be found and returned as documents.

// 7. Find the state with the highest unemployment rate. Hint. Use { \$limit: 1 }

The screenshot shows the MongoDB Compass interface with an aggregation pipeline applied to the 'Unemployment' collection. The pipeline consists of three stages: \$group, \$sort, and \$limit. The \$group stage calculates the average rate for each state. The \$sort stage sorts the results by avgRate in descending order. The \$limit stage returns only the top result. The output shows a single document for Arizona with an average rate of 9.274588477366255. A performance summary on the right indicates that the aggregation returned 47 documents and executed in 312 ms, while the initial collscan stage examined 885548 documents.

```
1 // **
2 * _id: The id of the group.
3 * fieldN: The first field name.
4 */
5 {
6   _id: "$State",
7   avgRate: {
8     $avg: "$Rate"
9   }
10 }
```

```
1 // **
2 * Provide any number of field/order pair
3 */
4 {
5   avgRate: -1
6 }
```

```
1 // **
2 * Provide the number of documents to lim
3 */
4 1
```

Performance Summary:

- GROUP**: Returned 47, Execution Time 312 ms
- COLLSCAN**: Returned 885548, Execution Time 343 ms, Documents Examined: 885548

The state with the highest unemployment rate is Arizona with an average rate of 9.27%. The aggregation first groups data by the state field and creates the average rate for each state in the dataset. In this stage, all the specific states and their respective average rates from the dataset can be found and returned as documents. Sort -1 is used to order the results from the State with the highest unemployment rate to the lowest. Finally, the aggregation of limit 1 only returns the one document with the highest unemployment rate, which is Arizona at 9.27%.

// 8. Count how many counties have an unemployment rate above 5%.

The screenshot displays the MongoDB Compass interface for a database named 'myMongoDB' and a collection named 'Unemployment'. The 'Aggregations' tab is active, showing a pipeline with three stages:

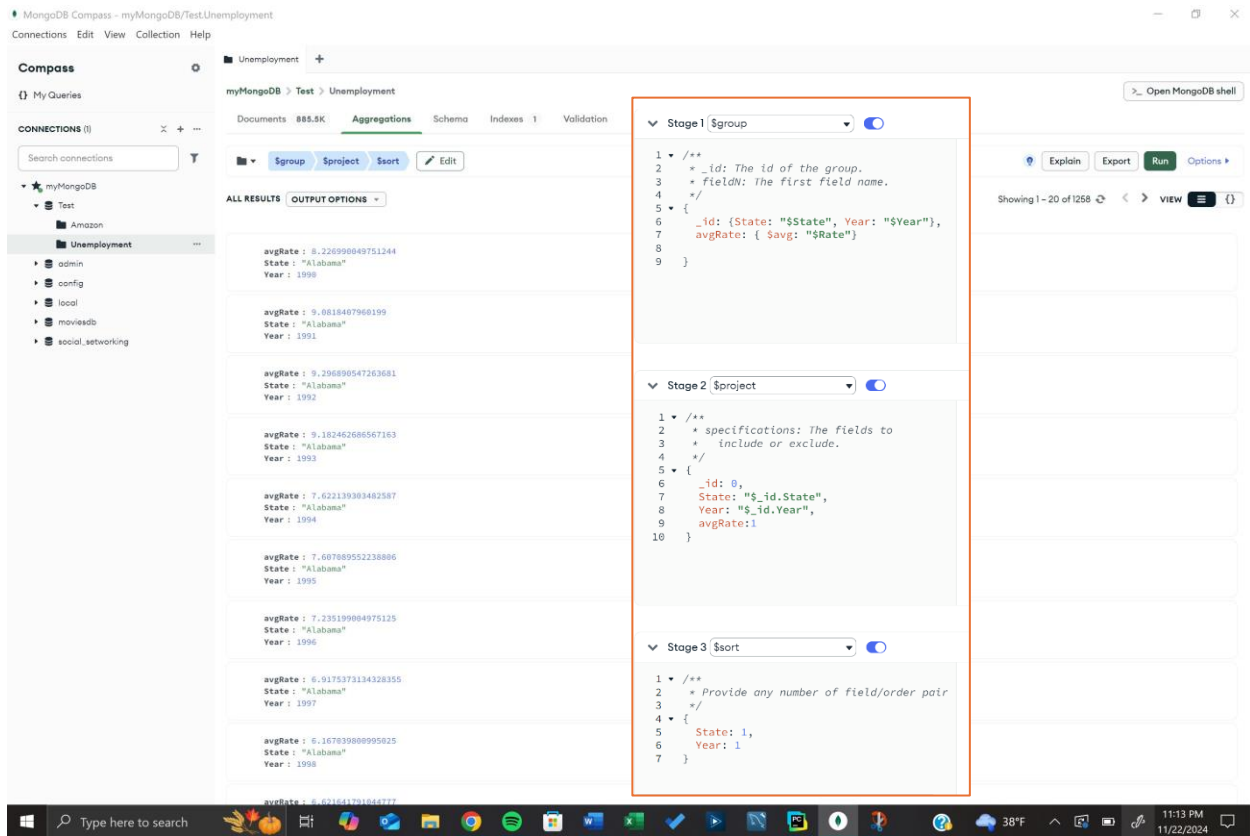
- Stage 1: \$group**  
1 `/**`  
2  `* id: The id of the group.`  
3  `* fieldN: The first field name.`  
4  `*/`  
5 `{`  
6  `_id: "$County",`  
7  `avgRate: {`  
8  `$avg: "$Rate"`  
9  `}`  
10 `}`
- Stage 2: \$match**  
1 `/**`  
2  `* query: The query in MQL.`  
3  `*/`  
4 `{`  
5  `avgRate: {`  
6  `$gt: 5`  
7  `}`  
8 `}`
- Stage 3: \$count**  
1 `/**`  
2  `* Provide the field name for the count.`  
3  `*/`  
4 `"Total Counties"`

The results pane shows 'Total Counties : 1238'. The Windows taskbar at the bottom indicates the system time is 10:44 PM on 11/22/2024.

There are 1238 counties with an unemployment rate above 5%. The aggregation groups data by the country and its average rate of unemployment over years data was collected. The aggregation then matches data by the field rate if it is greater than 5%. In this stage, all the specific matches to this query in the dataset can be found and returned as documents. This signifies they have an overall unemployment rate above 5%. In this stage, all the counties that match the filter in the dataset can be found and returned as documents. The final stage is count to determine the number of documents found in the previous stage. This is where it is determined that 1238 counties have an unemployment rate above 5%.



// 9. Calculate the average unemployment rate per state by year.



The screenshot shows the MongoDB Compass interface with an aggregation pipeline applied to the 'Unemployment' collection. The pipeline consists of three stages:

- Stage 1: \$group**

```
1 /**
2  * _id: The id of the group.
3  * fieldN: The first field name.
4  */
5 {
6   _id: {State: "$State", Year: "$Year"},
7   avgRate: { $avg: "$Rate" }
8 }
```
- Stage 2: \$project**

```
1 /**
2  * specifications: The fields to
3  * include or exclude.
4  */
5 {
6   _id: 0,
7   State: "$_id.State",
8   Year: "$_id.Year",
9   avgRate: 1
10 }
```
- Stage 3: \$sort**

```
1 /**
2  * Provide any number of field/order pair
3  */
4 {
5   State: 1,
6   Year: 1
7 }
```

The results are displayed in a table with columns for avgRate, State, and Year. The table shows 1258 documents, with the first few rows representing data for Alabama from 1998 to 2008.

There are 1258 documents stating the average unemployment rate per state by year. The aggregation groups data by state and year to calculate its average rate of unemployment. The aggregation then projects to show in the document the state and the given year's average rate of unemployment. If this was not added, then `_id` would read as object which can become confusing for the organization. To organize the documents, sort 1 was used for states to be in alphabetical order and their respective years in numerical order as well.