CN5006

Week 2 MONDODB AGG 1

pip 1

$match

/\*\*

\* query: The query in MQL.

\*/

{

  Education: "Master",

}

pip2

$group

/\*\*

\* \_id: The id of the group.

\* fieldN: The first field name.

\*/

{

  \_id: "$Marital Status",

  MinAge: {

    $min: "$Age",

  },

  AverageAge: {

    $avg: "$Age",

  },

  MaxAge: {

    $max: "$Age",

  },

  MinSalary: {

    $min: "$Salary",

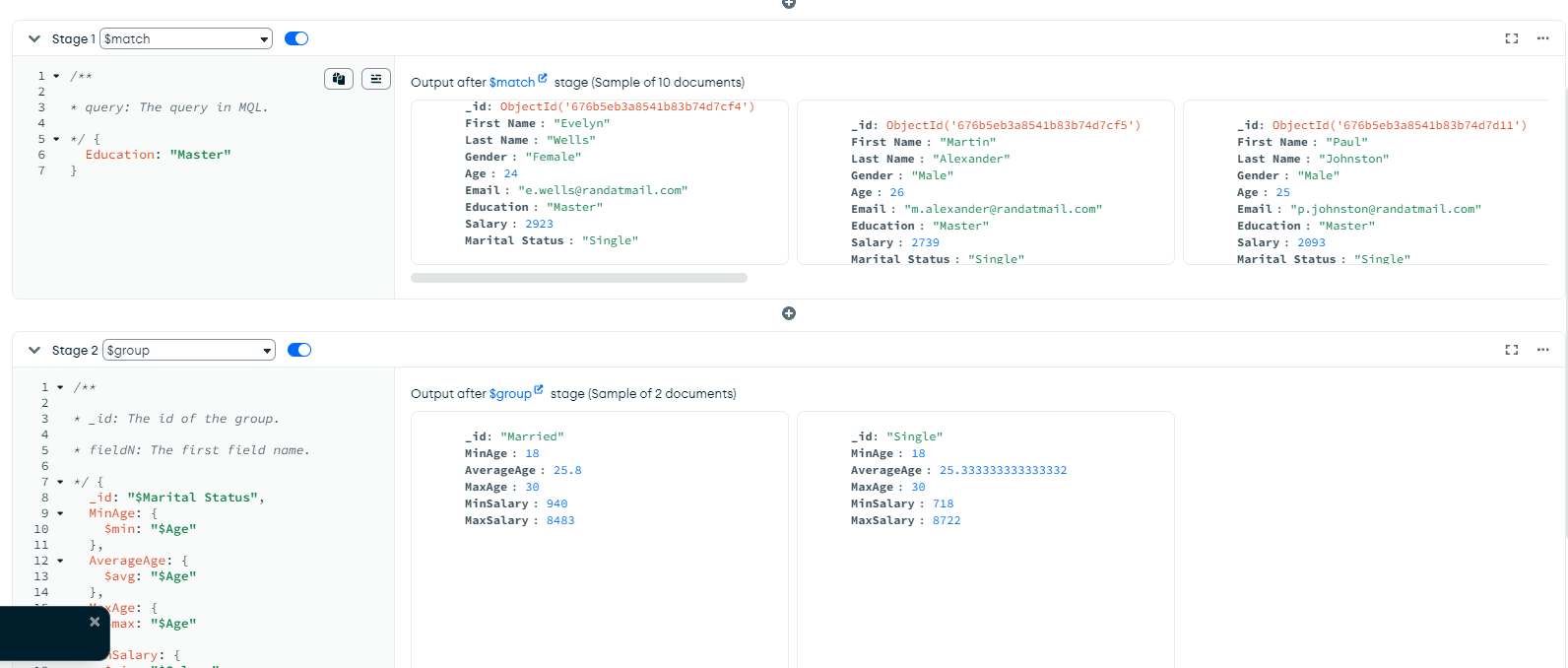
  },

  MaxSalary: {

    $max: "$Salary",

  },

}



This MongoDB aggregation pipeline has two stages: $match and $group. The $match stage filters the data to only include records where the Education field is "Master." Once the filtering is done, the $group stage organizes the remaining records by Marital Status, creating groups like "Single" or "Married." For each group, it calculates the minimum, average, and maximum values of Age and Salary. In simple terms, this pipeline helps summarize the data about people with a Master's degree by marital status, providing insights like the youngest, oldest, and average age or salary in each group.

AGG 2

Pip 1

/\*\*

\* query: The query in MQL.

\*/

{

Gender: "Female" // Replace "Gender" and "Female" with your actual field names and values

}

Pip 2

/\*\*

\* \_id: The id of the group.

\* fieldN: The first field name.

\*/

{

\_id: "$Age",

// Group by the "Age" field

MinSalary: {

$min: "$Salary"

},

// Calculate the minimum salary in each age group

MaxSalary: {

$max: "$Salary"

},

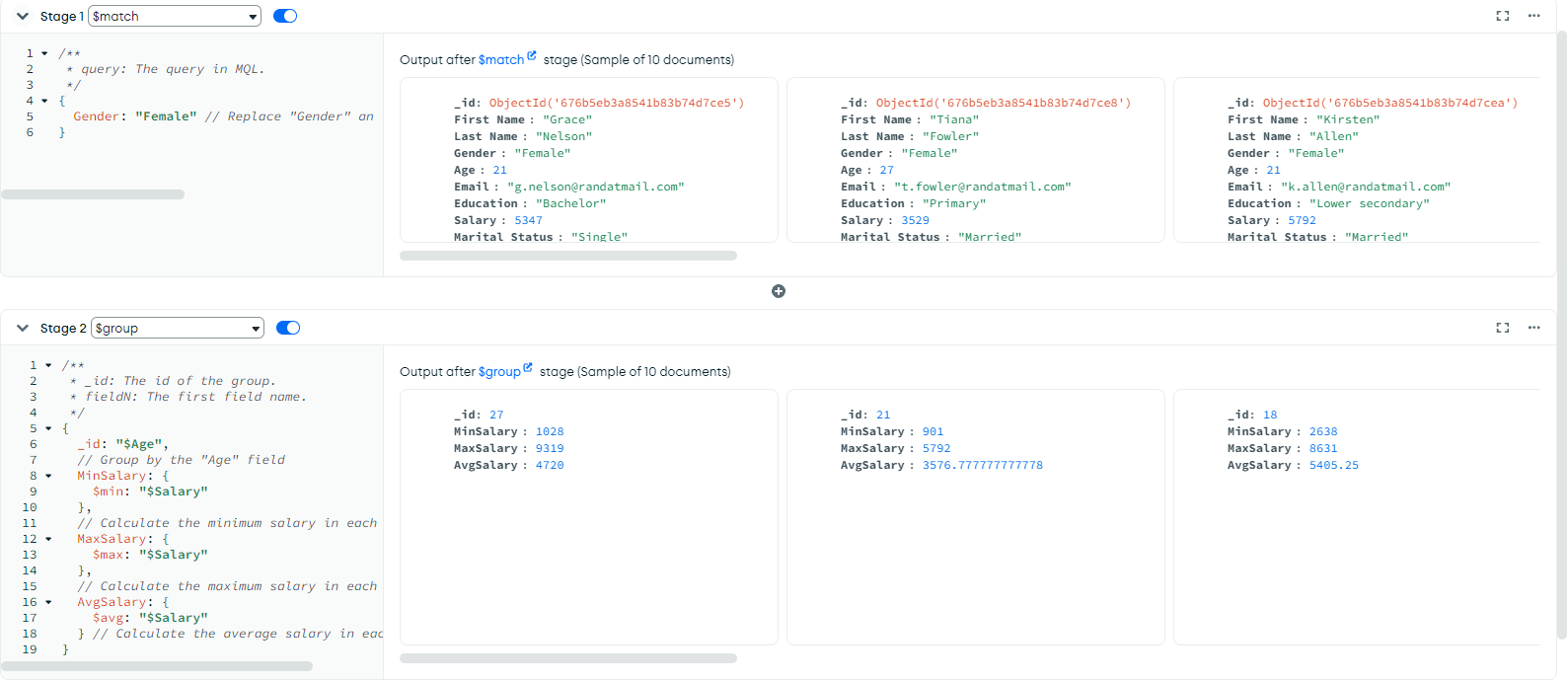
// Calculate the maximum salary in each age group

AvgSalary: {

$avg: "$Salary"

} // Calculate the average salary in each age group

}



This MongoDB aggregation query helps find the minimum, maximum, and average salary for each age group among females. It starts with the $match stage, which filters the data to include only records where the Gender is "Female." Next, the $group stage groups the data by Age and calculates the minimum salary ($min), maximum salary ($max), and average salary ($avg) for each age group. Finally, the $sort stage organizes the results by age in ascending order, making it easier to read. In short, this code summarizes salary statistics for females by their age.

AGG 3

Pip 1

/\*\*

\* query: The query in MQL.

\*/

{

Gender: "Male" // Replace "Gender" and "Female" with your actual field names and values

}

Pip 2

/\*\*

\* \_id: The id of the group.

\* fieldN: The first field name.

\*/

{

\_id: "$Age",

// Group by the "Age" field

MinSalary: {

$min: "$Salary"

},

// Calculate the minimum salary in each age group

MaxSalary: {

$max: "$Salary"

},

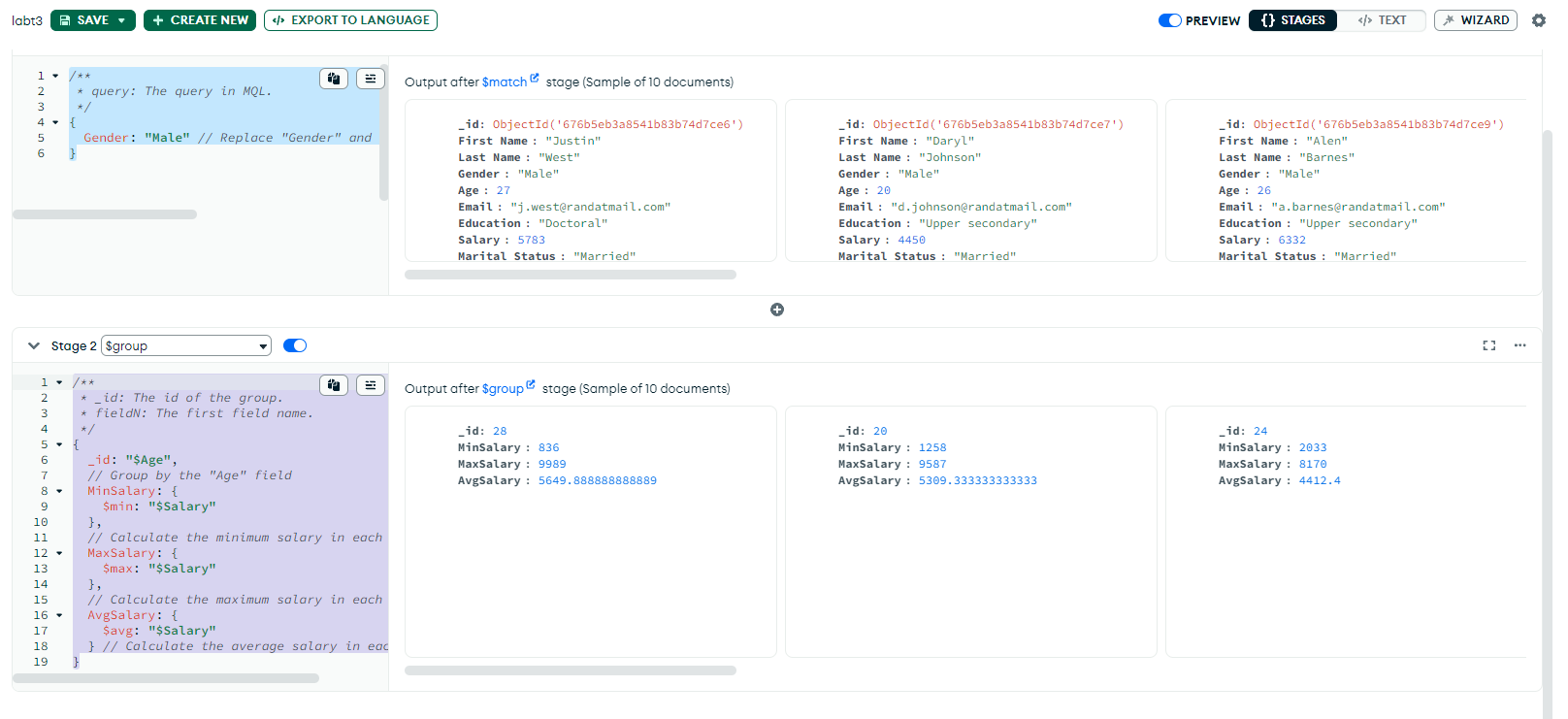
// Calculate the maximum salary in each age group

AvgSalary: {

$avg: "$Salary"

} // Calculate the average salary in each age group

}

This follows AGGREGATE 2, the only change is on the first pip where instead of ”Female” it is ”Male”.

AGG 4

Pip 1

/\*\*

\* \_id: The id of the group.

\* fieldN: The first field name.

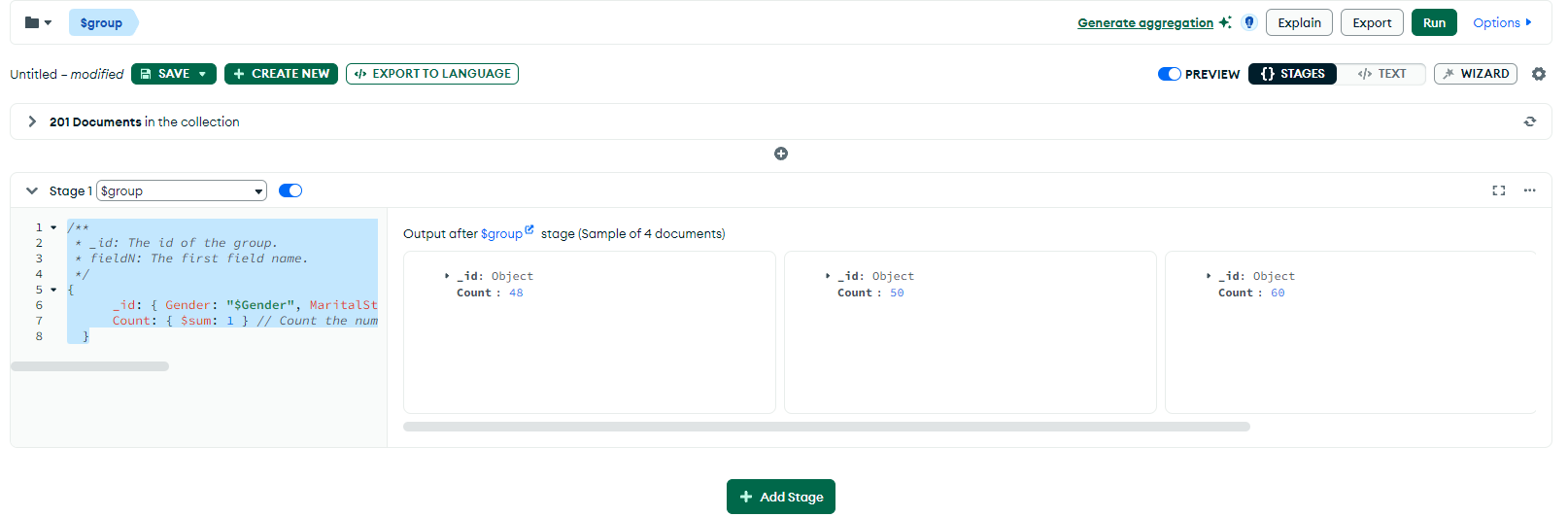
\*/

{

\_id: { Gender: "$Gender", MaritalStatus: "$Marital Status" }, // Group by Gender and Marital Status

Count: { $sum: 1 } // Count the number of documents in each group

}



The screenshot shows the result of a $group stage in MongoDB, where the data is being grouped, but the \_id field is shown as "Object" rather than specific values like "Female, Married" or "Male, Single." This typically happens when the query is grouping by a combination of fields, resulting in a more complex structure. If you group by both Gender and Marital Status, you should expect four distinct groups in the output: "Female, Married", "Female, Single", "Male, Married", and "Male, Single." Each of these groups will have a corresponding count that represents the number of records in that category. For example, if the output shows Count: 48 for a group, it means that there are 48 records in that group (e.g., 48 married females). The issue in the screenshot is likely due to the grouping not being specified clearly for these two individual fields, leading to the generic "Object" being displayed instead of the expected group names.

REFLECTION

In this lab task I have learned how to use MONGODB COMPASS application. I have learned how to connect to local database host, then import csv file into it. With this information, I have then learned how to create AGGREGATIONS, where it returns specific outputs depending on the requirements I have set on the pipeline. I have also learned on how to insert data into the database manually, how to edit and delete any of the data that is on the database.