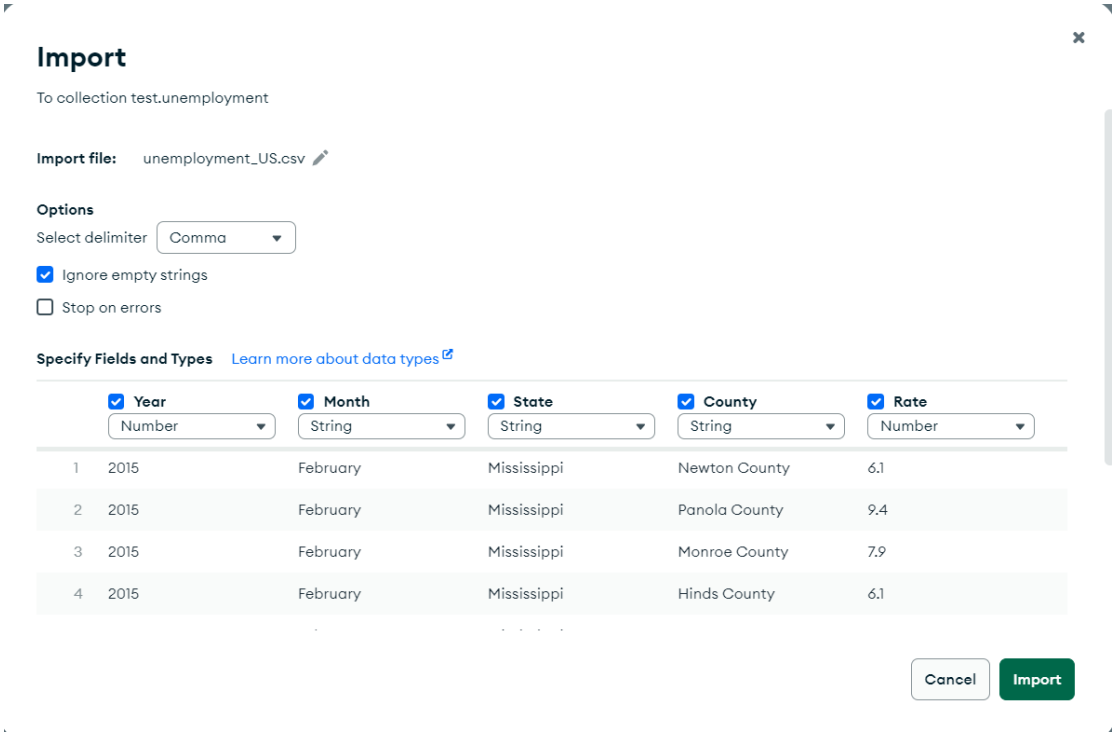
- **Title:** DB Assignment 5
- **Your Name:** Deare
- **Date:** 2024/11/11

Step1:

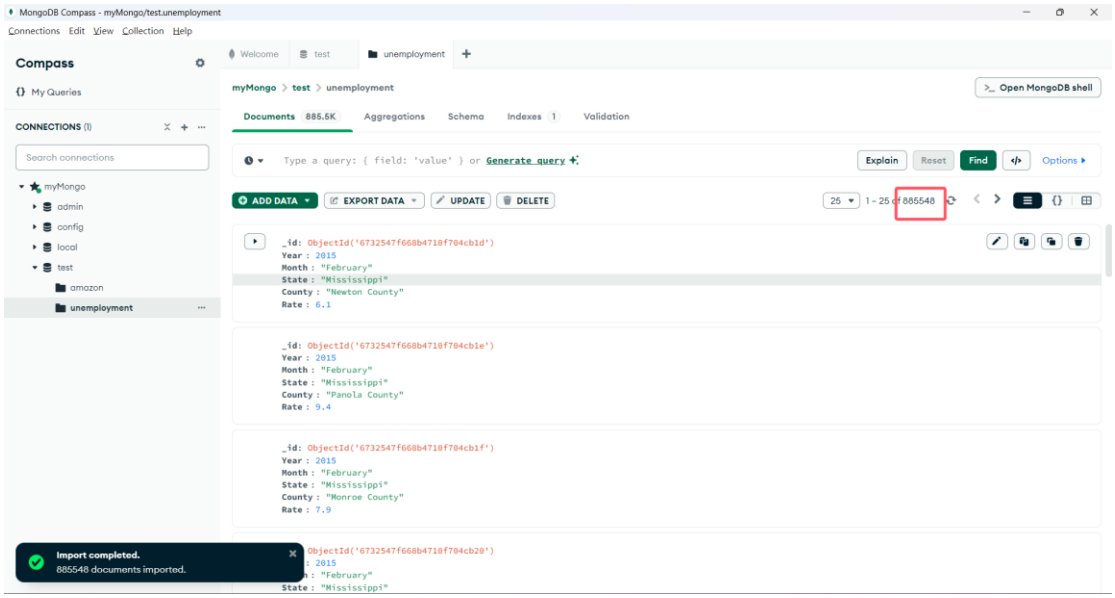
- Created the collection **unemployment** in the **test** database and populated it with the downloaded CSV data.

Step 2:

- Import Year and **Rate fields as Number type, Month, State and Country fields as String type.**



- Successfully imported 885548 documents



Step 3:

- Analyze the schema:
  - Click on the 'Schema' tab and select 'Analyze Schema' to view the results of the schema analysis.
  - **Collection Name:** unemployment
  - **Sample Size:** The analysis is based on a sample of 1000 documents.

  Field Analysis:

  1. _id (ObjectId)

     Description: This field automatically generated by MongoDB serves as the **primary key for the collection**. It uniquely identifies each document in the collection.
  2. County (String)

     Distribution: The graph shows a variety of counties represented with varying frequencies.

     Analysis: The distribution suggests a balanced representation of data across different counties with 1% frequencies and Chariton Country, Mahaska Country, and Jackson Country have the highest 2% frequencies which is useful for regional analysis.
  3. Month (String)

     Distribution: Months are represented with varying frequencies.

     Analysis: January has height frequencies with a percentage of 15%, and September has the lowest frequencies with a percentage of 4%.
  4. Rate (Double)

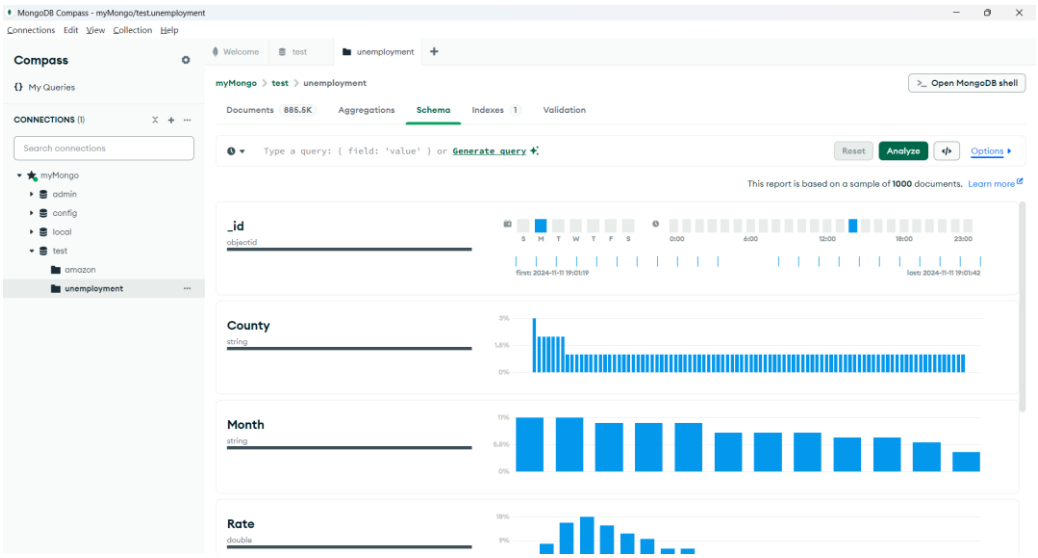
     Distribution: This field shows a rate and with its percentage, likely represents unemployment rates.

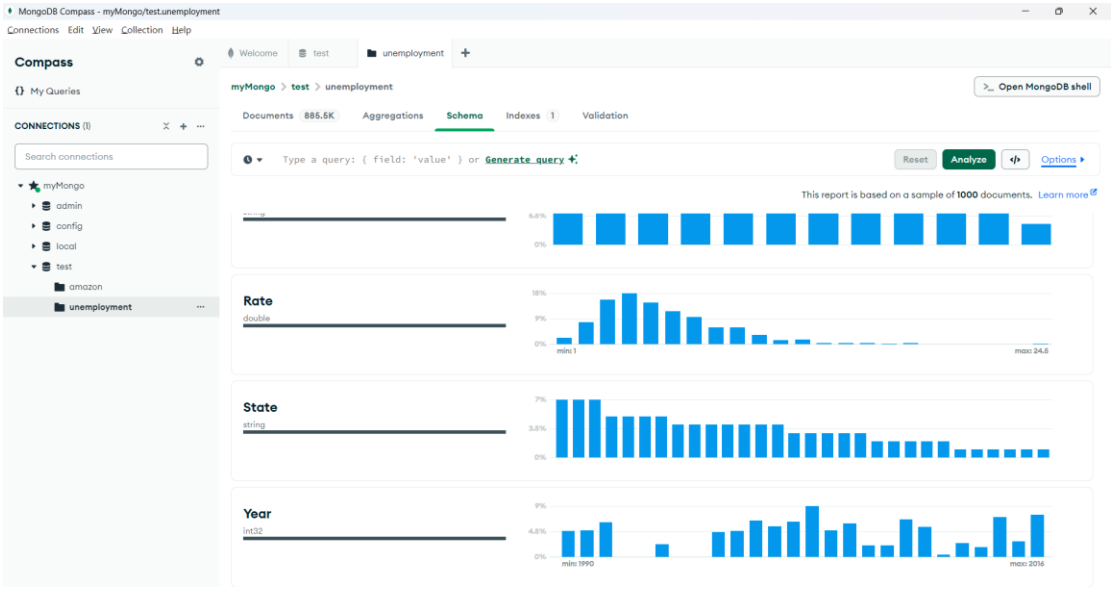
     Analysis: *Under min and max for Rate:* Min: 0.7 - This represents the lowest value found in the Rate field across the sampled documents. Max: 34.6 - This represents the highest value found in the Rate field.
  5. State (String)

     Analysis: Graph distribution indicates that the data encompasses a variety of states. Missouri has the highest frequency with a percentage of 8%, "New Mexico", "Wyoming", "Washington", "California", "Rhode Island", "New York" and "Utah" has the lowest frequency with a percentage of 1%.
  6. Year (Int32)

     Range: Min: 1990, Max: 2015

     Analysis: Reflecting the years from 1990 to 2015 across the sampled documents. 2010-2011 has the highest frequencies with percentage of 7%. Year fields could be useful for trend analysis over time.





Step 4:

Queries in **MongoDB Shell**:

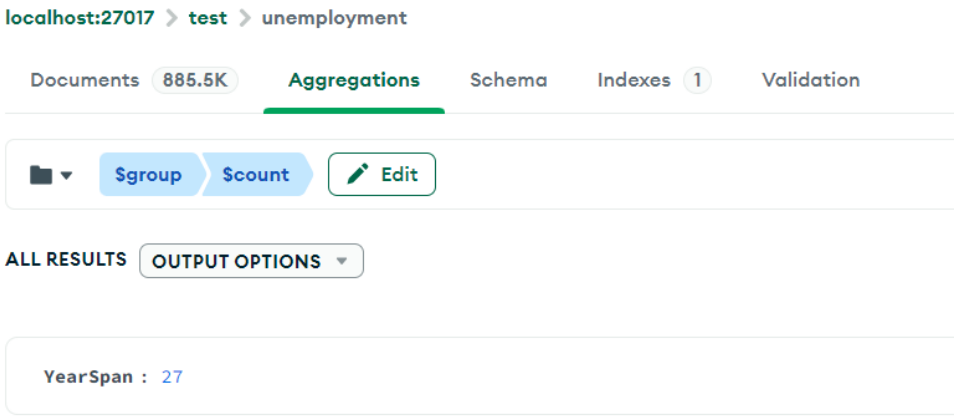
1. Over how many years was the unemployment data collected?

Means: the total span of years during which the unemployment data was gathered. This involves determining the range of years covered in the dataset.

I am going to use aggregation group by year fields and use count to count year spans.

```
db.unemployment.aggregate([
{
  $group:
    /**
     * _id: The id of the group.
     * fieldN: The first field name.
     */
```

```
        {
          _id: "$Year"
        }
    },
    {
      $count:
        /**
         * Provide the field name for the count.
         */
        "YearSpan"
    });
```



]

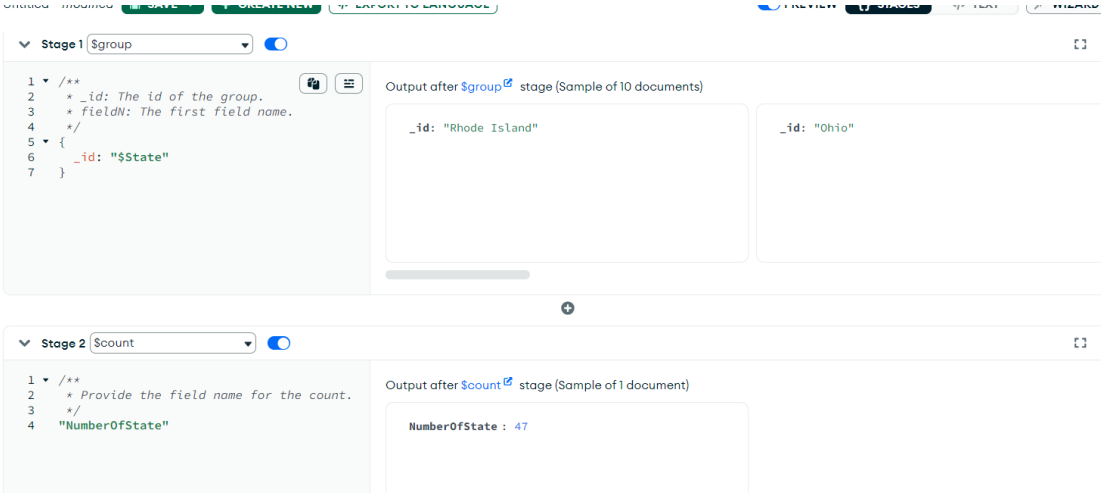
Data on unemployment rates were collected over 27 years.

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

Assuming its meaning is that this question asked for a distinct state name and its total number.

A solution similar to question 1, this time going to group by State fields and count the state in an aggregate query.

```
db.getCollection('unemployment').aggregate(
  [
    { $group: { _id: '$State' } },
    { $count: 'NumberOfState' }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```



3. What does this query compute?

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

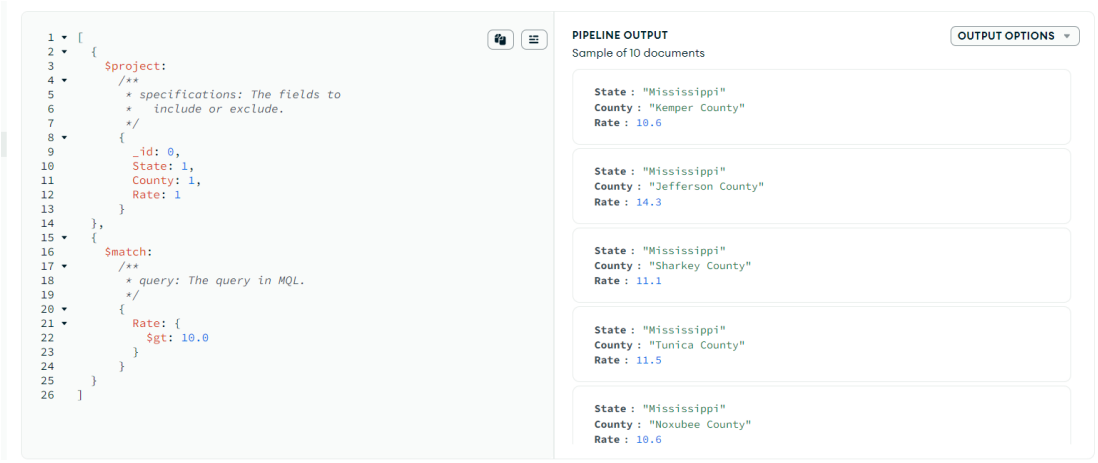
   This query means that retrieved unemployment people who have a Rate less than 1.0 and then count their number.

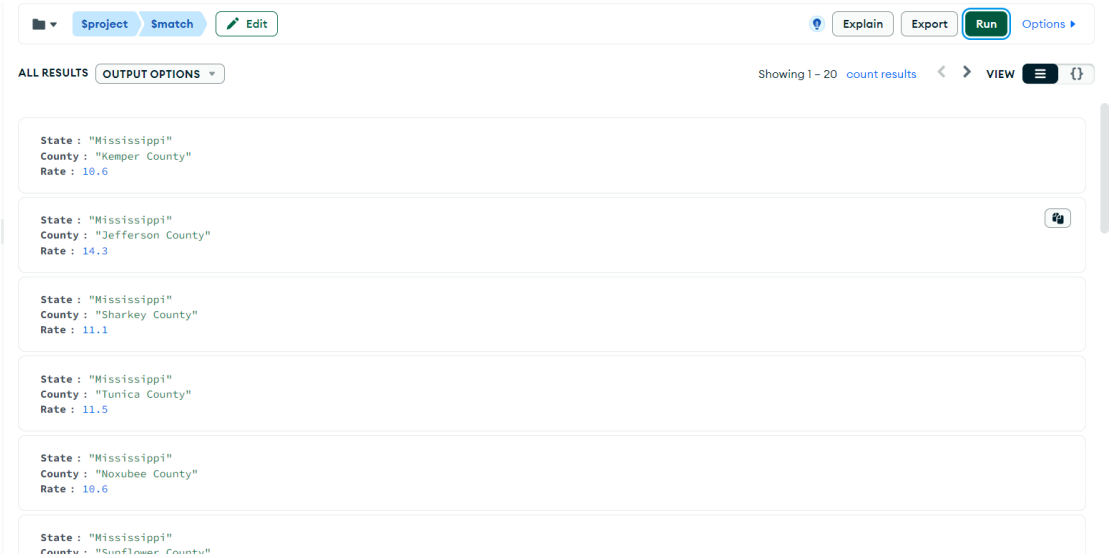
```
> db.unemployment.find({Rate : {$lt: 1.0}}).count()
< 657
```

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

Assuming this question asks for a filter for states with an unemployment rate higher than 10%, which I think means retrieving values greater than 10.0 in the Rate field.

```
db.getCollection('unemployment').aggregate(
  [
    {
      $project: {
        _id: 0,
        State: 1,
        County: 1,
        Rate: 1
      }
    },
    { $match: { Rate: { $gt: 10 } } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

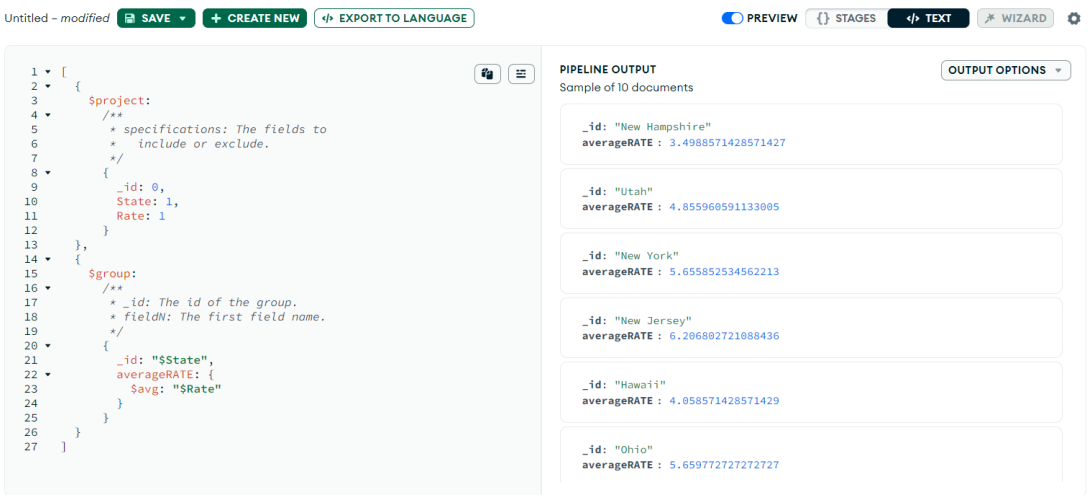




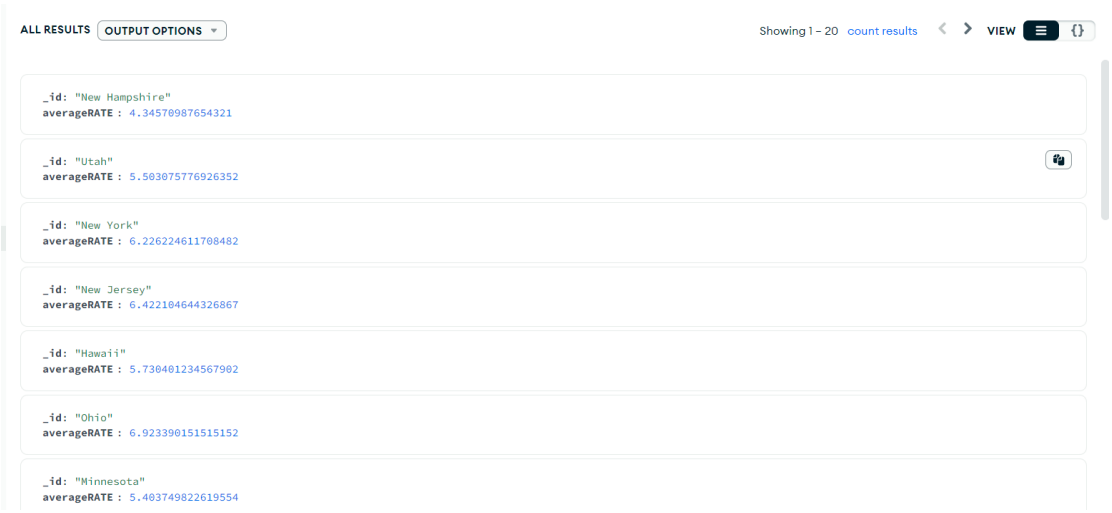
5. **Calculate the average unemployment rate across all states.**
   Assuming the result includes all the state names and its **average unemployment rates.**

```
db.getCollection('unemployment').aggregate(
  [
    { $project: { _id: 0, State: 1, Rate: 1 } },
    {
      $group: {
        _id: '$State',
        averageRATE: { $avg: '$Rate' }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

   This is sample size's result:



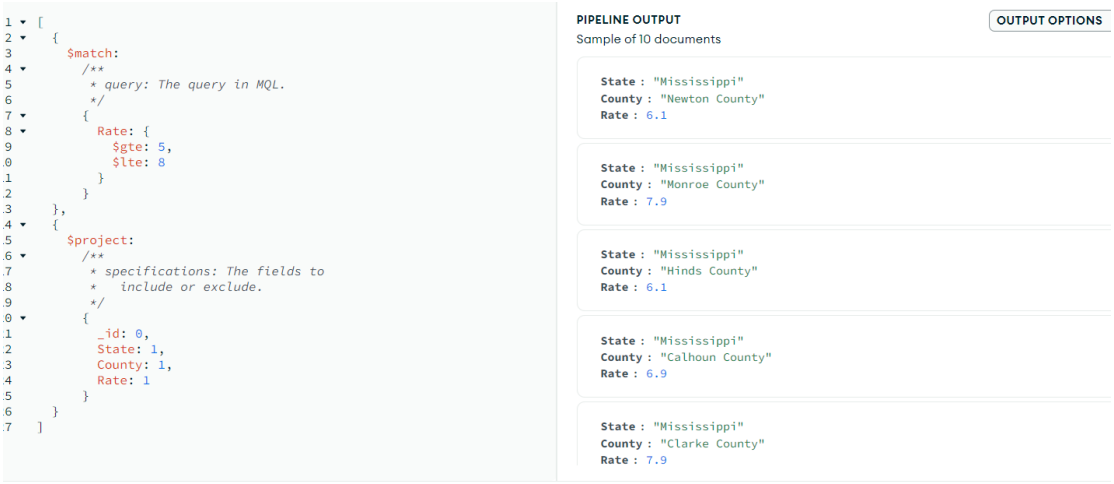
This is after clicking run button to get all the result:



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

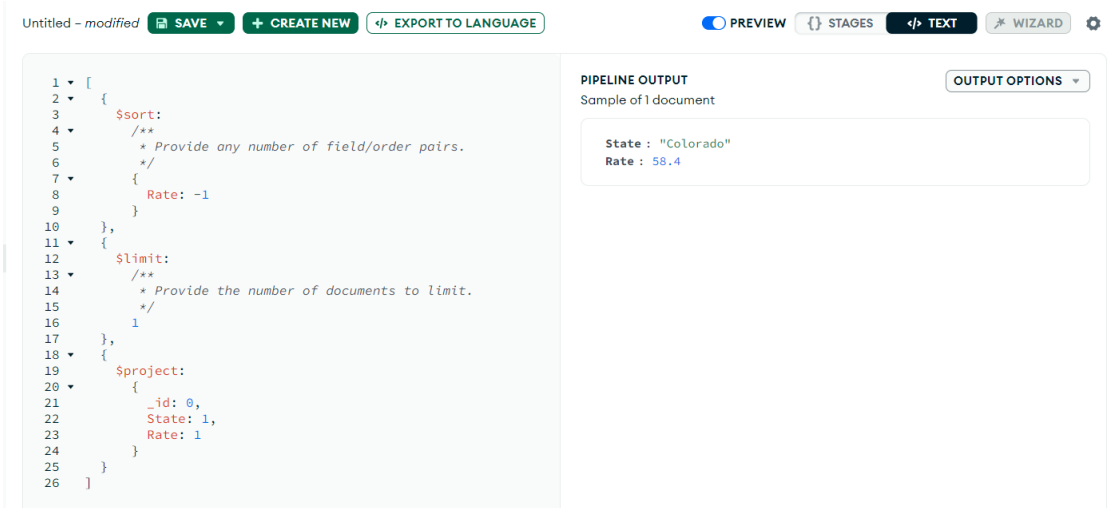
I assume this question is asked about all state names and county names where the unemployment rate values between 5.0 and 8.0.

```
db.getCollection('unemployment').aggregate(
  [
    { $match: { Rate: { $gte: 5, $lte: 8 } } },
    {
      $project: {
        _id: 0,
        State: 1,
        County: 1,
        Rate: 1
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

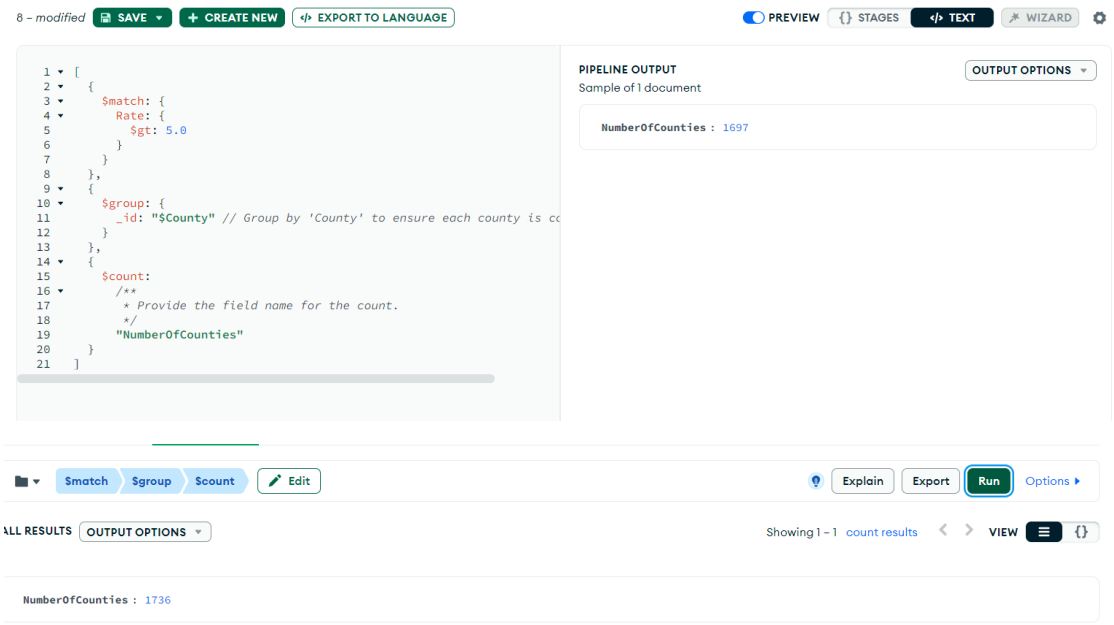


7. Find the state with the **highest unemployment rate**. **Hint. Use { $limit: 1 }**

```
db.getCollection('unemployment').aggregate(
  [
    { $sort: { Rate: -1 } },
    { $limit: 1 },
    { $project: { _id: 0, State: 1, Rate: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true });
```



8. Count how many counties have an unemployment rate above **5%**.
Assuming this question is about asking unemployment Rate value greater than 5.0.

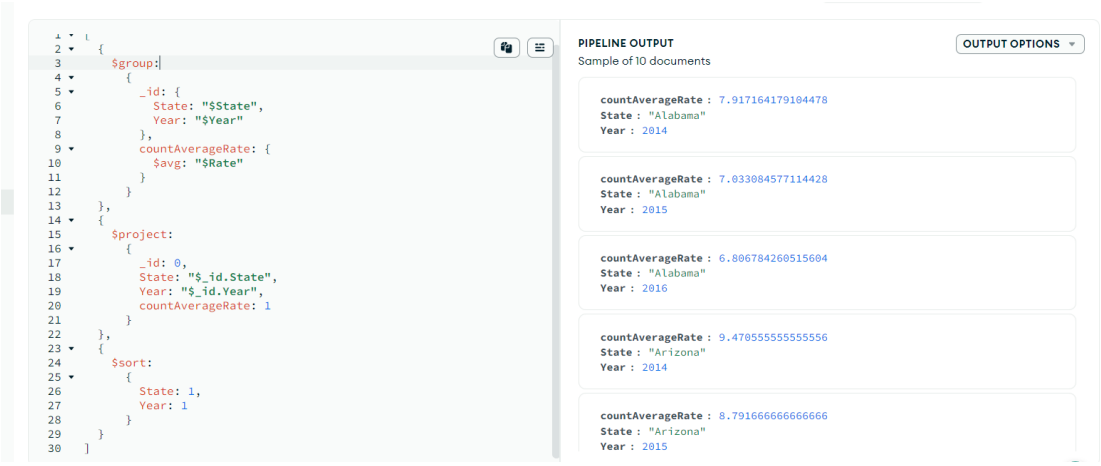



```
db.getCollection('unemployment').aggregate(
  [
    { $match: { Rate: { $gt: 5 } } },
    { $group: { _id: '$County' } },
    { $count: 'NumberOfCounties' }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

9. Calculate the average unemployment rate per state by year.
The aggregation pipeline would involve grouping the data first by state and year, calculating the average rate for these groups, and then projecting the results in a readable format for example I will sort by state and then year for easier reading.

```
db.getCollection('unemployment').aggregate(
  [
    {
      $group: {
        _id: { State: '$State', Year: '$Year' },
        countAverageRate: { $avg: '$Rate' }
      }
    },
```

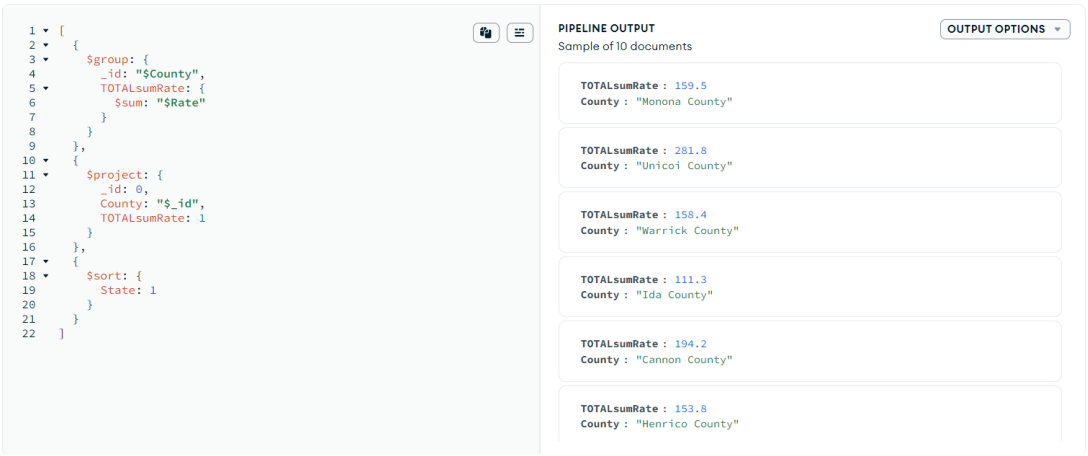
```
    {
      $project: {
        _id: 0,
        State: '$_id.State', // Map 'State' from the group '_id'
        Year: '$_id.Year', // Map 'Year' from the group '_id'
        countAverageRate: 1
      }
    },
    { $sort: { State: 1, Year: 1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

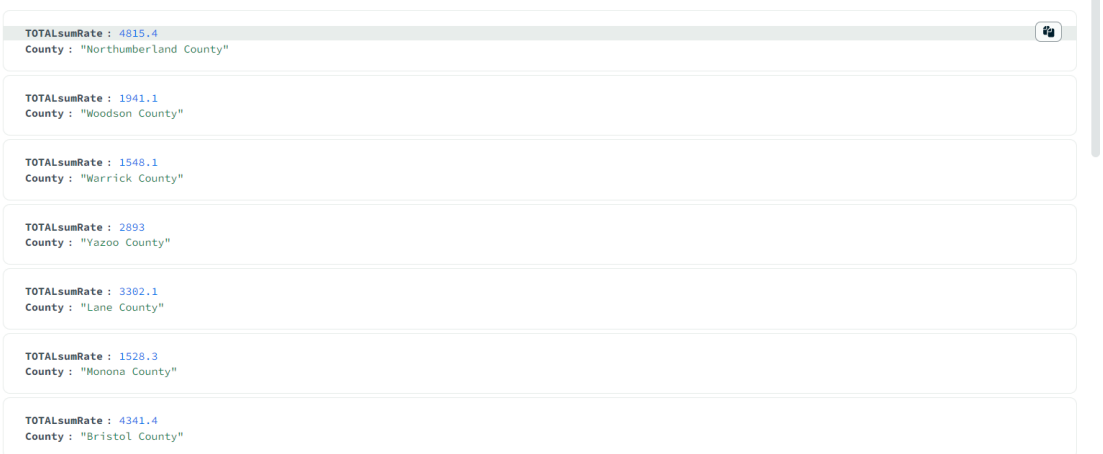


10. (Extra Credit) For each state, calculate the total unemployment rate across all counties (sum of all county rates)
    Use an aggregation pipeline that groups the data by state, sums the unemployment rates for all counties within each state, and then projects the results in a readable format, and then sorts the result by state name in alphabetical order, which is not a requirement, however, I want to add this extra step for my result more readable.

```
db.getCollection('unemployment').aggregate(
  [
    {
      $group: {
        _id: '$County',
        TOTALsumRate: { $sum: '$Rate' }
      }
    },
    {
      $project: {
        _id: 0,
        County: '$_id',
```

```
            TOTALsumRate: 1
          }
        },
        { $sort: { State: 1 } }
      ],
      { maxTimeMS: 60000, allowDiskUse: true }
    );
```





11. (Extra Credit) The same as Query 10 but for states with data from 2015 onward
    The output from this pipeline will be documents where each document represents a state, along with the total summed unemployment rate of all its counties from 2015 onward, and then the result by state name in alphabetical order, which is not a requirement, however, I want to add this extra step for my result more readable.

```
db.getCollection('unemployment').aggregate(
  [
    { $match: { Year: { $gt: 2015 } } },
    {
      $group: {
        _id: '$State',
```

```
            TOTALsumRateAFTER2015: { $sum: '$Rate' }
        }
    },
    {
        $project: {
            _id: 0,
            State: '$_id',
            TOTALsumRateAFTER2015: 1
        }
    },
    { $sort: { State: 1 } }
],
{ maxTimeMS: 60000, allowDiskUse: true }
);
```