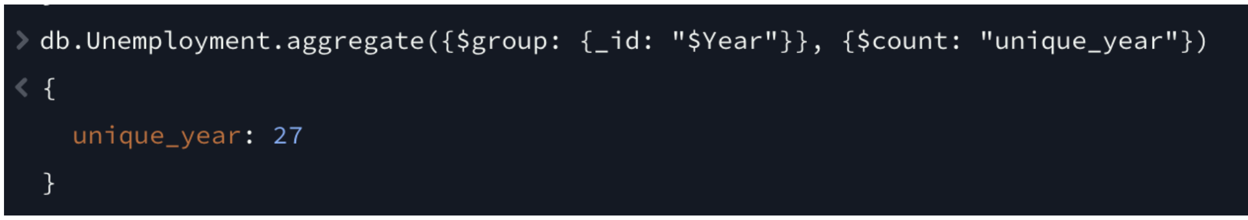
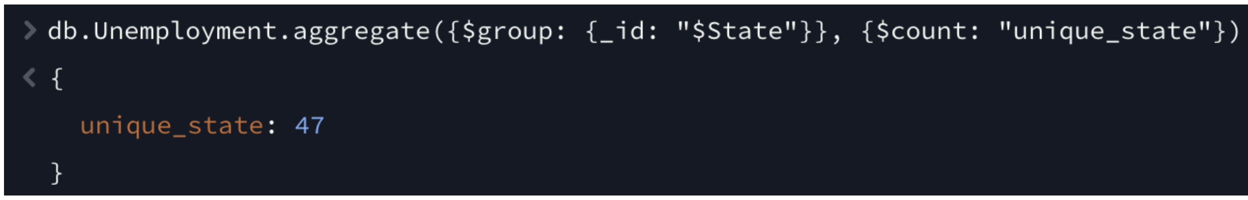
DB Assignment 5
Brittany Klose
11/21/24

1. **Over how many years was the unemployment data collected?**
    a. For this query I used group and count. First I grouped the data by year. The group aggregate in Mongo will grab a distinct year. Then I used count to find the total number of years the unemployment data was collected.
    b. 

```
> db.Unemployment.aggregate({$group: {_id: "$Year"}}, {$count: "unique_year"})
< {
    unique_year: 27
  }
```

2. **How many states were reported on in this dataset?**
    a. I approached this query as I did the previous using group and count to find distinct states and then count the number of unique states included in the dataset.
    b. 

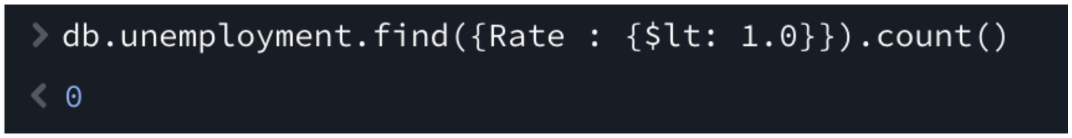
```
> db.Unemployment.aggregate({$group: {_id: "$State"}}, {$count: "unique_state"})
< {
    unique_state: 47
  }
```

3. **What does this query compute?**

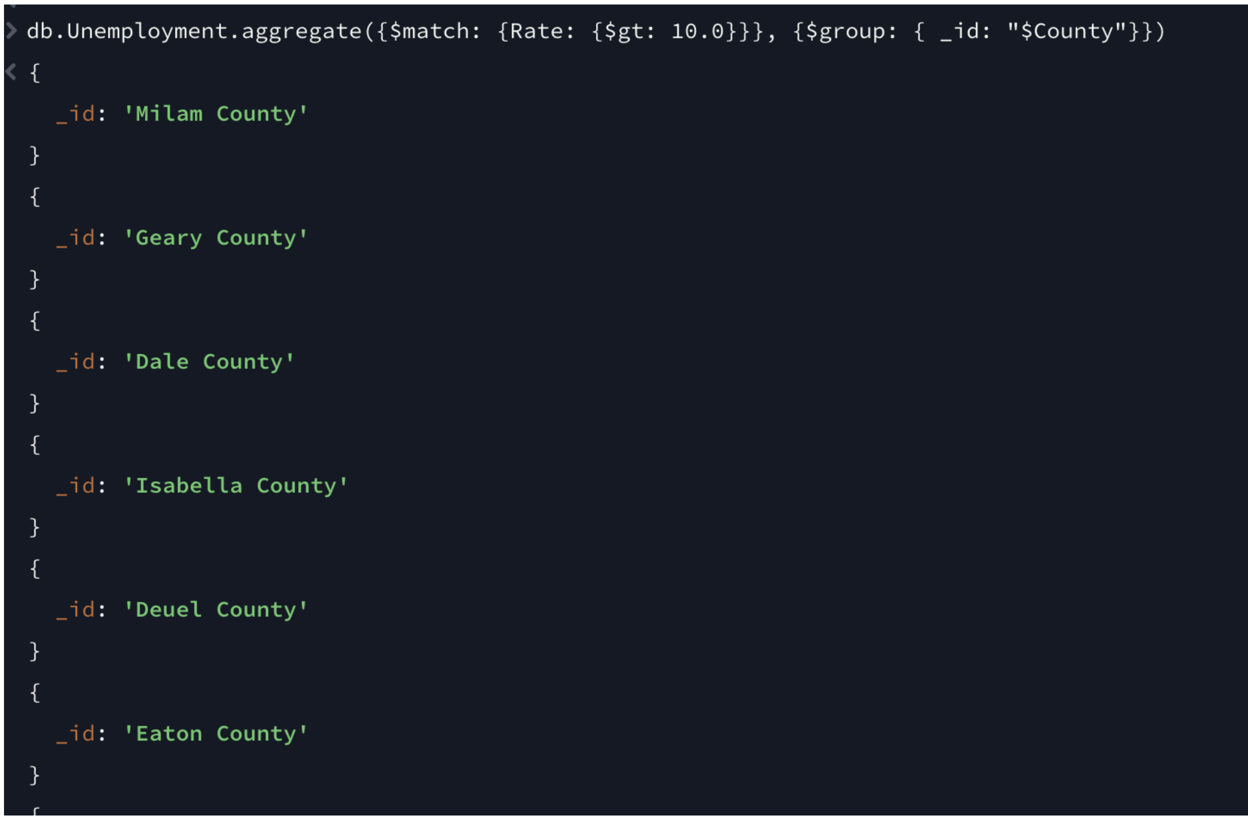
    db.unemployment.find({Rate : {$lt: 1.0}}).count()

    a. When I put this query into my mongo shell I received 0 as seen below. It looks like the query is trying to select the Rate field where the rate is less than 1 and then count the number of rates that were less than 1. I tried the query using 6 instead of 1 as I knew for sure there was at least 1 however

I still got 0 as the result. Looks like the query would need to be adjusted to properly work.

b. 

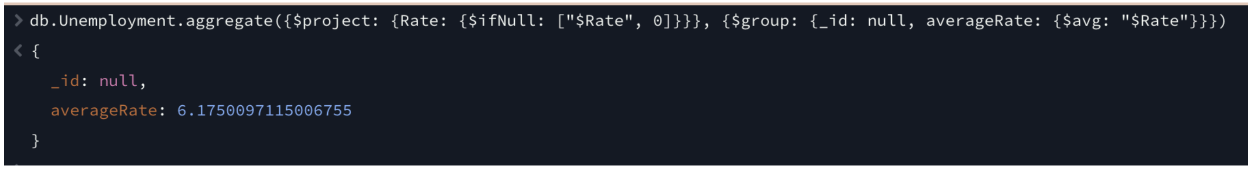
```
> db.unemployment.find({Rate : {$lt: 1.0}}).count()
< 0
```

4. **Find all counties with unemployment rate higher than 10%**
   a. For this query I used match and group. First I used match to locate rows where the rate field was greater than 10% with $gt. Then I used the group aggregate to select distinct counties. A snippet of the first 6 results are below.
   b. 

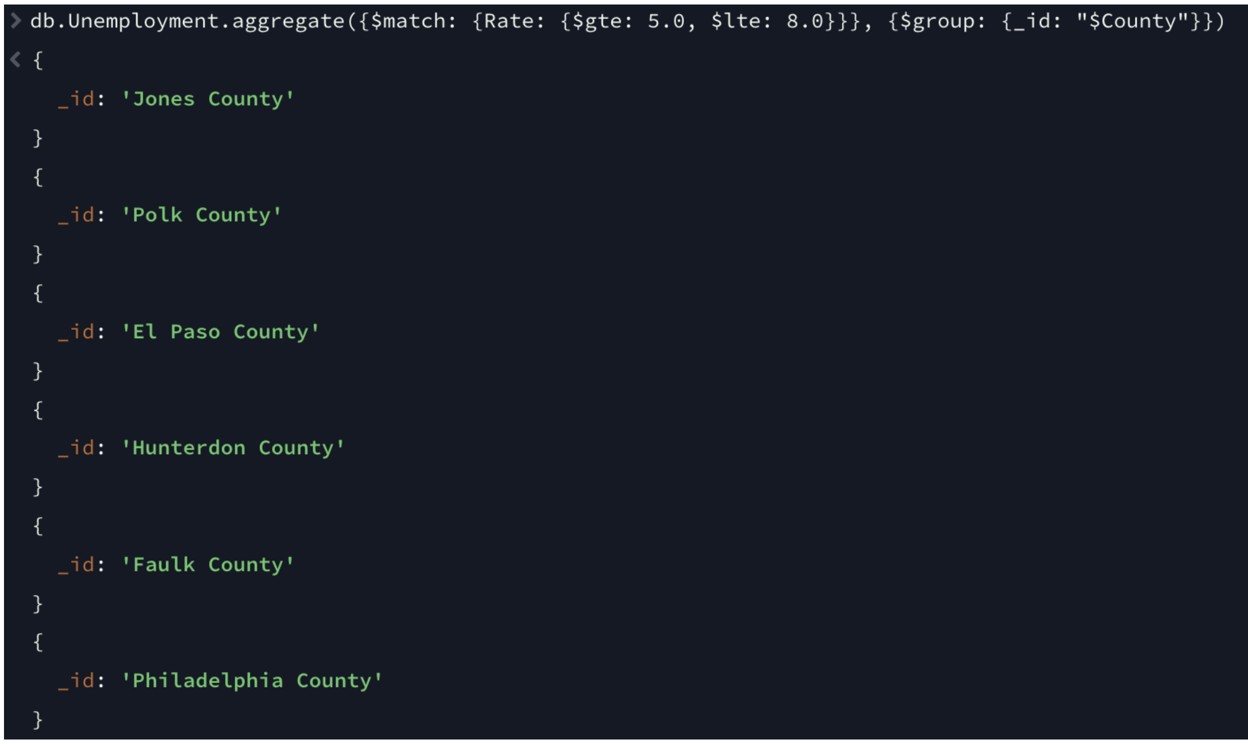
```
> db.Unemployment.aggregate({$match: {Rate: {$gt: 10.0}}}, {$group: { _id: "$County"}})
< {
    _id: 'Milam County'
  }
  {
    _id: 'Geary County'
  }
  {
    _id: 'Dale County'
  }
  {
    _id: 'Isabella County'
  }
  {
    _id: 'Deuel County'
  }
  {
    _id: 'Eaton County'
  }
```

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

a. For this query I used the aggregates project and group to find the average rate. First I used Project to replace any null rates with 0. Then I used Group to collect all rates and find the average.



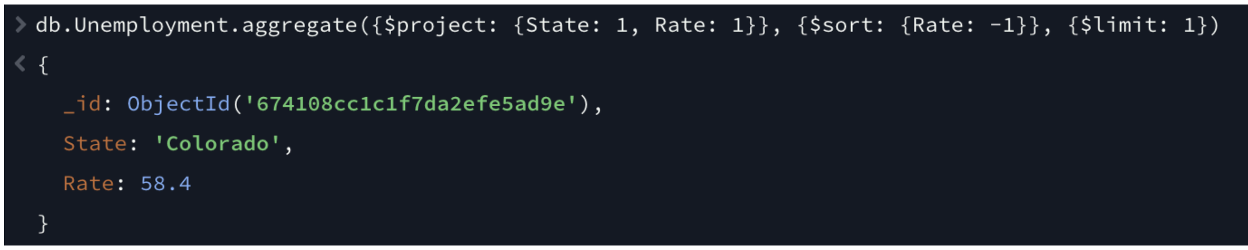
```
> db.Unemployment.aggregate({$project: {Rate: {$ifNull: ["$Rate", 0]}}}, {$group: {_id: null, averageRate: {$avg: "$Rate"}}})
< {
    _id: null,
    averageRate: 6.1750097115006755
  }
```

6. **Find all counties with an unemployment rate between 5% and 8%.**
   a. This query required using Match to find a rate that was greater than or equal to 5% and less than or equal to 8%. And then Group to select the distinct counties. The results below show the first 6 rows.
   b. 

```
> db.Unemployment.aggregate({$match: {Rate: {$gte: 5.0, $lte: 8.0}}}, {$group: {_id: "$County"}})
< {
    _id: 'Jones County'
  }
  {
    _id: 'Polk County'
  }
  {
    _id: 'El Paso County'
  }
  {
    _id: 'Hunterdon County'
  }
  {
    _id: 'Faulk County'
  }
  {
    _id: 'Philadelphia County'
  }
```

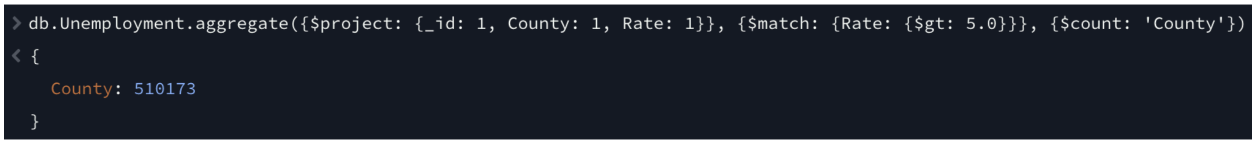
7. **Find the state with the highest unemployment rate. Hint. Use { $limit: 1 }**
   a. To answer this query I used Project, Sort, and Limit. I used Project to grab the state name and rate, sort to arrange the states and rates in order of

highest to lowest, and the limit to only select the state with the highest rate.

b. 

```
> db.Unemployment.aggregate({$project: {State: 1, Rate: 1}}, {$sort: {Rate: -1}}, {$limit: 1})
< {
    _id: ObjectId('674108cc1c1f7da2efe5ad9e'),
    State: 'Colorado',
    Rate: 58.4
  }
```

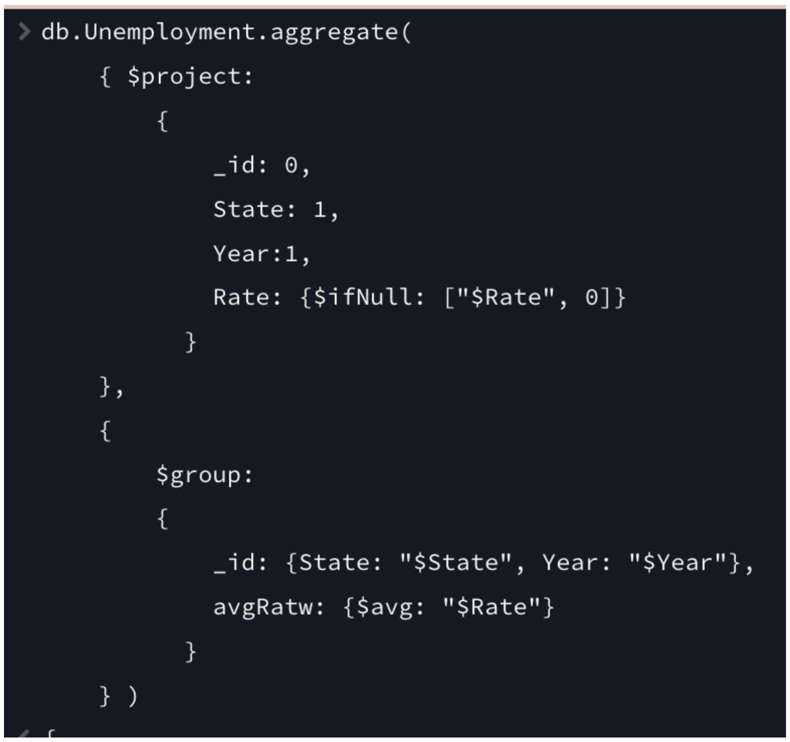
8. **Count how many counties have an unemployment rate above 5%.**
   a. For this query I used Match to find counties with rates greater than 5%. And then I used Count to find the number of counties.



```
> db.Unemployment.aggregate({$project: {_id: 1, County: 1, Rate: 1}}, {$match: {Rate: {$gt: 5.0}}}, {$count: 'County'})
< {
    County: 510173
  }
```

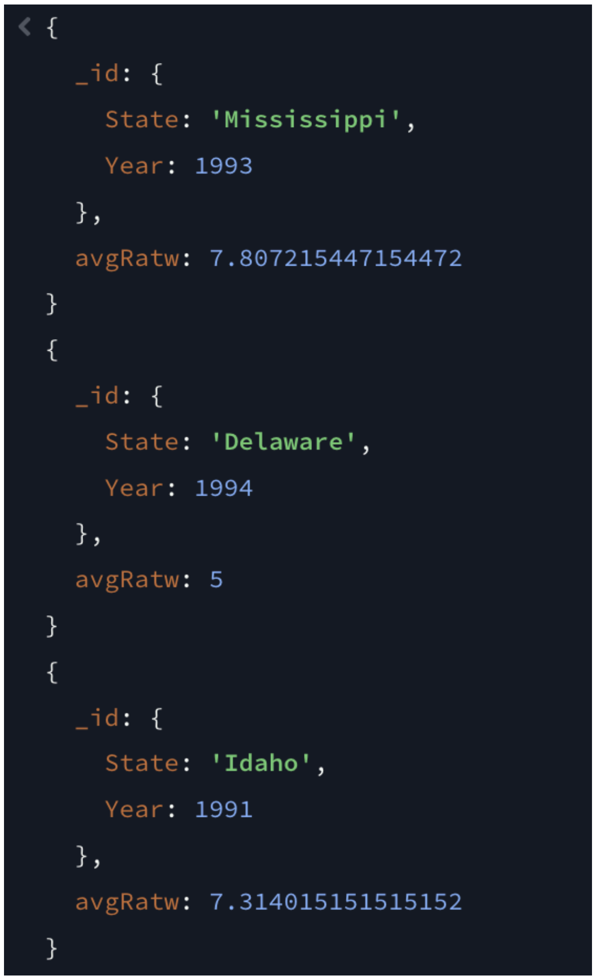
9. **Calculate the average unemployment rate per state by year.**
   a. To get this query I used Project and Group. With the Project query I selected to include the state, year, and rate. I added in ‘ifnull’ to ensure if any rates were null they would still be part of the query and counted as 0. Lastly I used the group aggregate to group by state and year and calculate the average unemployment rate. Below includes my query and a snippet of the results.

b.



```
> db.Unemployment.aggregate(
    { $project:
        {
            _id: 0,
            State: 1,
            Year:1,
            Rate: {$ifNull: ["$Rate", 0]}
          }
    },
    {
        $group:
        {
            _id: {State: "$State", Year: "$Year"},
            avgRatw: {$avg: "$Rate"}
          }
    } )
```

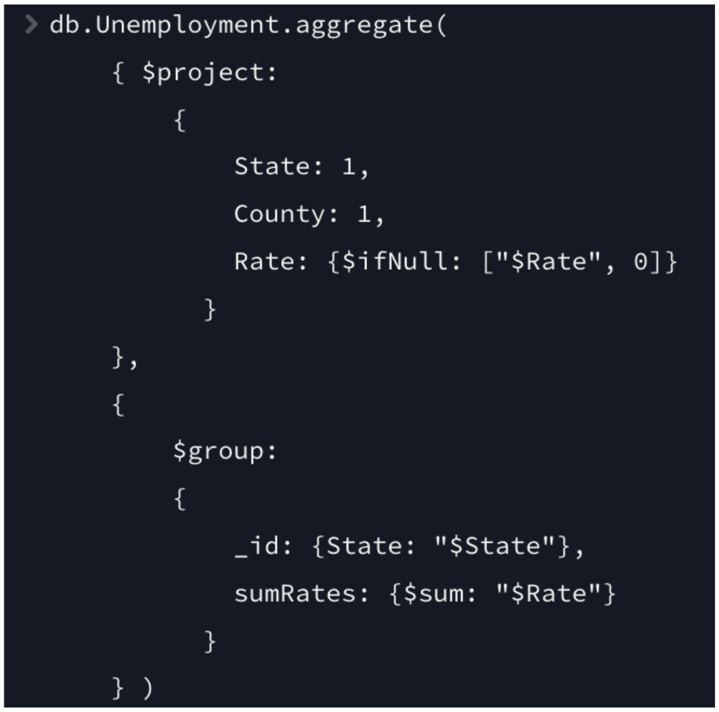
c.



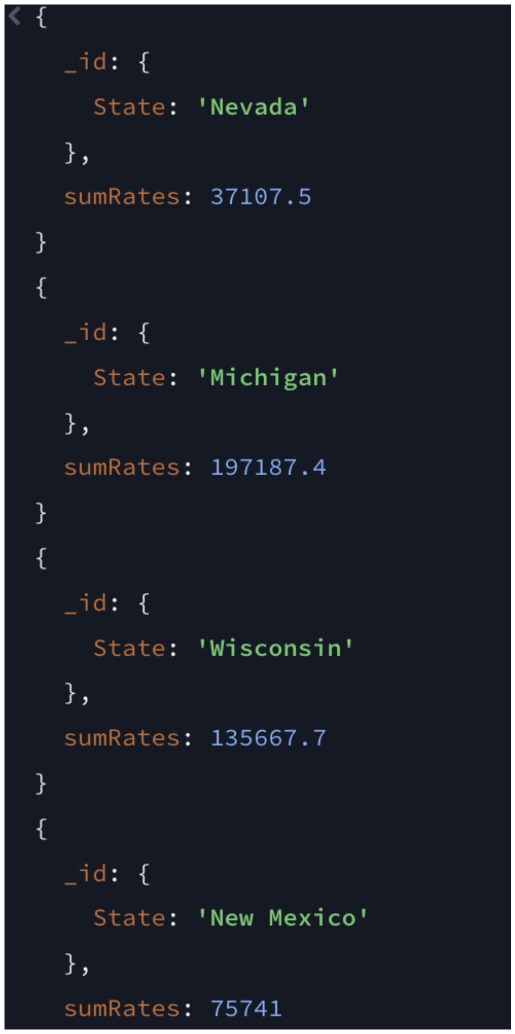
```
< {
    _id: {
      State: 'Mississippi',
      Year: 1993
    },
    avgRatw: 7.807215447154472
  }
  {
    _id: {
      State: 'Delaware',
      Year: 1994
    },
    avgRatw: 5
  }
  {
    _id: {
      State: 'Idaho',
      Year: 1991
    },
    avgRatw: 7.314015151515152
  }
```

10. **(Extra Credit) For each state, calculate the total unemployment rate across all counties (sum of all county rates).**

    a. Similarly to the previous problem I performed the project and group aggregates to execute this query. With Project I set which fields to include; which were state, county, and rate. For rate I again defined any null values as 0 so they will be part of the end sum. In the second part I used Group to group by state and calculate the sum of unemployment rates for each state. Below includes my query and a snippet of the results.



```
> db.Unemployment.aggregate(
      { $project:
          {
              State: 1,
              County: 1,
              Rate: {$ifNull: ["$Rate", 0]}
            }
      },
      {
          $group:
          {
              _id: {State: "$State"},
              sumRates: {$sum: "$Rate"}
            }
      } )
```



```
< {
    _id: {
      State: 'Nevada'
    },
    sumRates: 37107.5
  }
  {
    _id: {
      State: 'Michigan'
    },
    sumRates: 197187.4
  }
  {
    _id: {
      State: 'Wisconsin'
    },
    sumRates: 135667.7
  }
  {
    _id: {
      State: 'New Mexico'
    },
    sumRates: 75741
```