HW 5

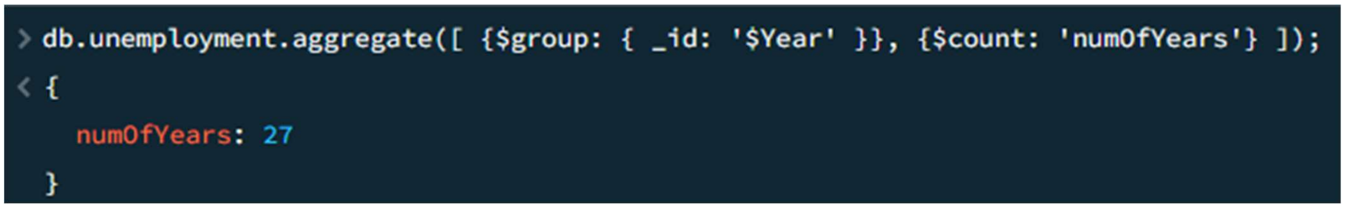
Title: DB Assignment 5

Name: Brennen Cramp

Date: 11/22/2024

Problem 1:

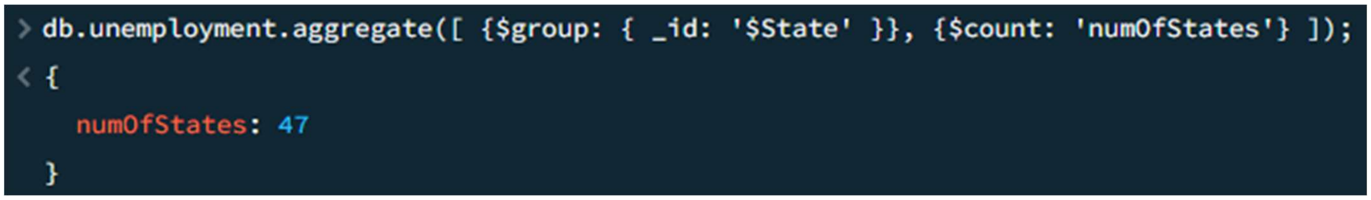
To find how many years the unemployment data was collected, I created an aggregate query. I started with a group stage where I grouped by the year field value, making each year distinct. From here, I utilized the count stage which counts the documents returned from the previous stage and returned this value as “numOfYears” which is 27 since the years spanned from 1990 to 2016.



```
> db.unemployment.aggregate([ {$group: { _id: '$Year' }}, {$count: 'numOfYears'} ]);
< {
    numOfYears: 27
  }
```

Problem 2:

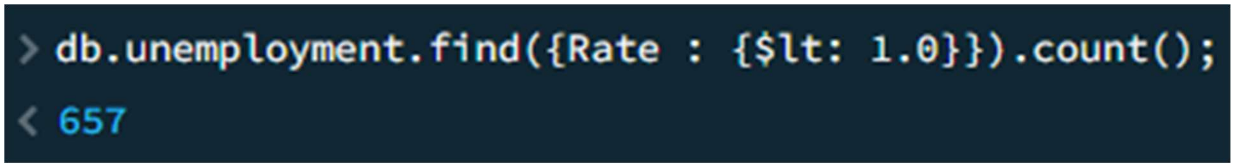
To find how many states were reported on in this dataset, I created an aggregate query. I started with a group stage where I grouped by the state field value, making each state distinct. From here, I utilized the count stage which counts the documents returned from the previous stage and returned this value as “numOfStates”.



```
> db.unemployment.aggregate([ {$group: { _id: '$State' }}, {$count: 'numOfStates'} ]);
< {
    numOfStates: 47
  }
```

Problem 3:

The query db.unemployment.find({Rate : {$lt: 1.0}}).count() computes the number of documents that have an unemployment rate of less than 1.0% over the entire collection period.



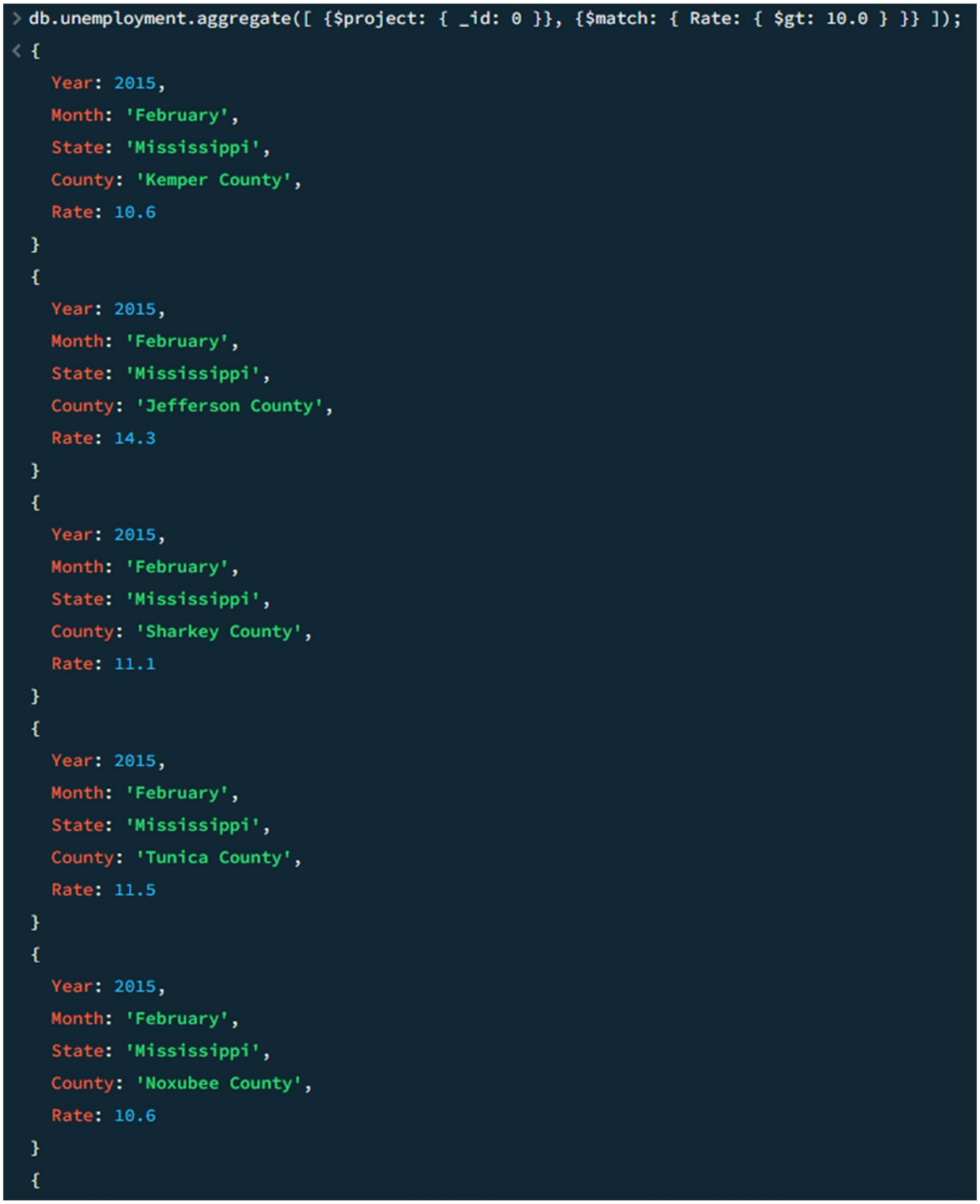
```
> db.unemployment.find({Rate : {$lt: 1.0}}).count();
< 657
```

Problem 4:

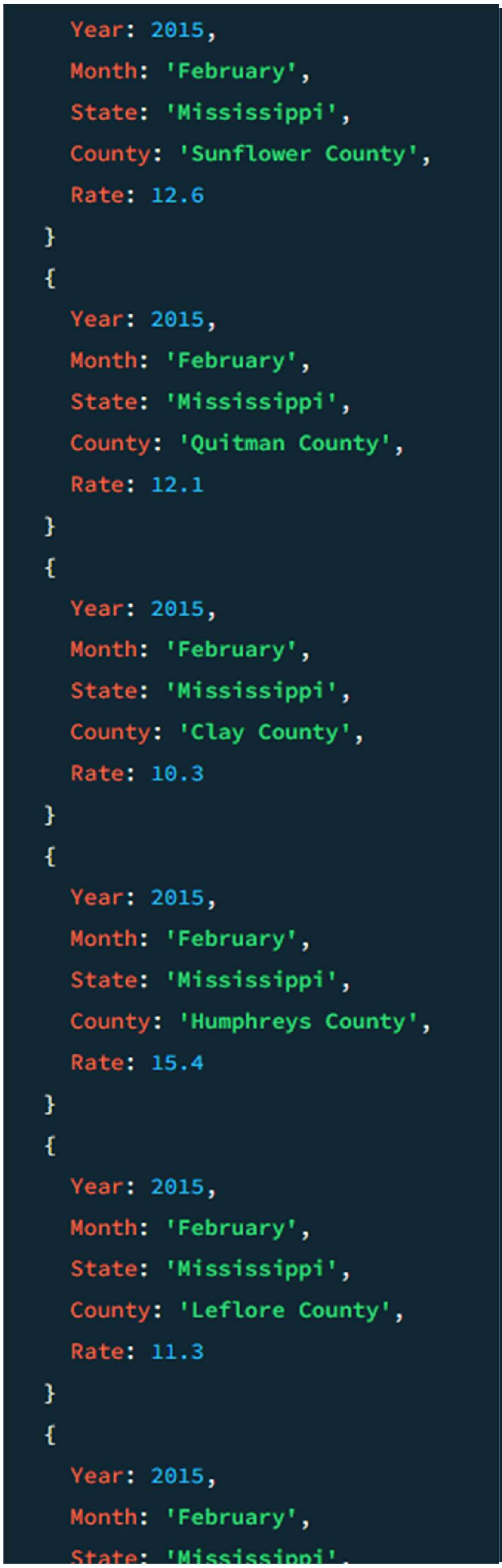
To find all counties where the unemployment rate is greater than 10%, I created an aggregate query. I started with a project stage to get rid of the "_id" field for the output where I then used a match stage to filter all the documents where the rate is greater than 10.0%. I also used two assumptions for the problem:

- We can assume that the question is asking monthly and NOT an average all time OR per year.
- We can assume that the question is asking for the output to be a list of counties (including relevant data) and NOT the size of the list.

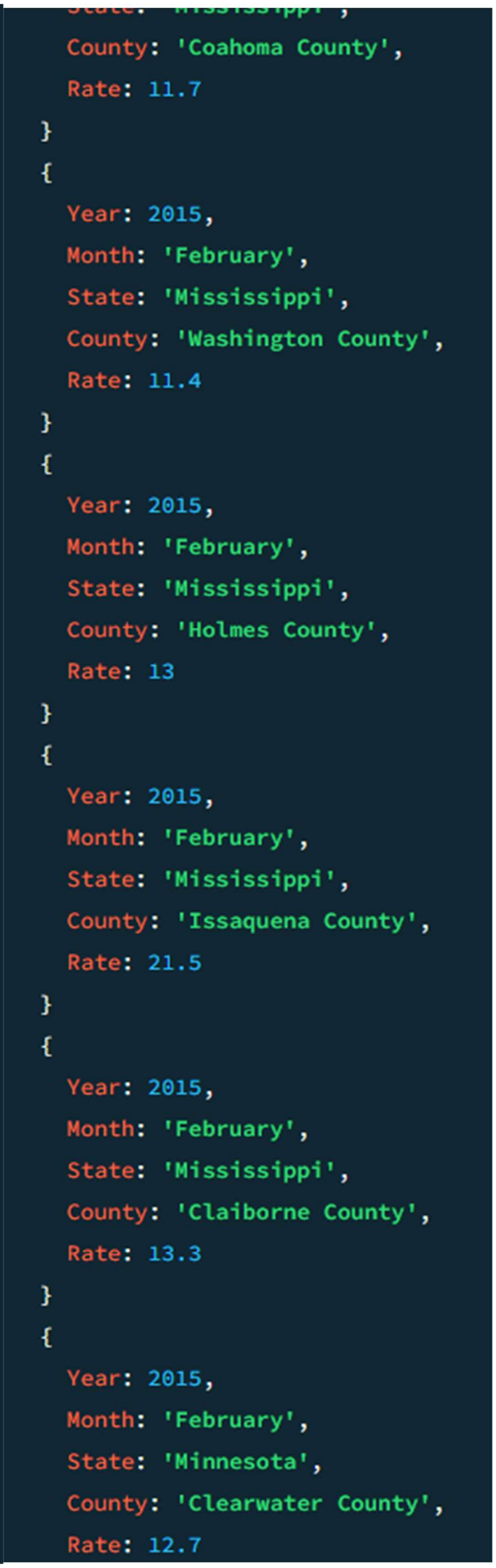
Pictures of the output are shown below.



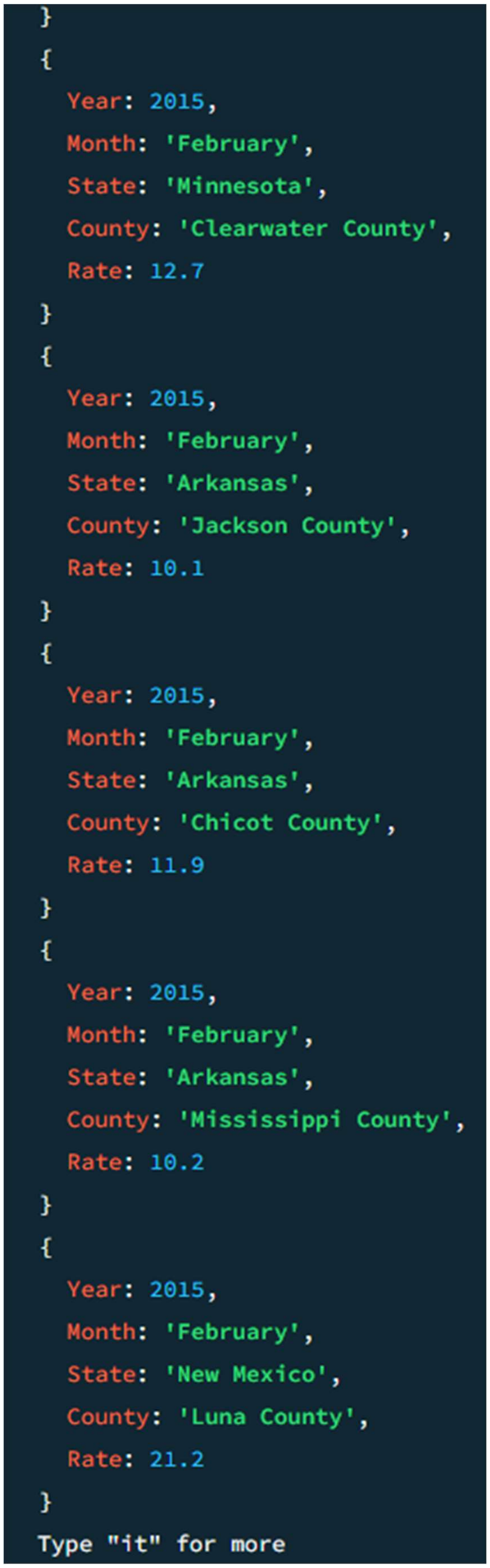
```
> db.unemployment.aggregate([ {$project: { _id: 0 }}, {$match: { Rate: { $gt: 10.0 } }} ]);
< {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Kemper County',
    Rate: 10.6
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Jefferson County',
    Rate: 14.3
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Sharkey County',
    Rate: 11.1
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Tunica County',
    Rate: 11.5
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Noxubee County',
    Rate: 10.6
  }
  {
```



```
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Sunflower County',
  Rate: 12.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Quitman County',
  Rate: 12.1
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Clay County',
  Rate: 10.3
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Humphreys County',
  Rate: 15.4
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Leflore County',
  Rate: 11.3
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
```



```
  State: 'Mississippi',
  County: 'Coahoma County',
  Rate: 11.7
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Washington County',
  Rate: 11.4
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Holmes County',
  Rate: 13
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Issaquena County',
  Rate: 21.5
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Claiborne County',
  Rate: 13.3
}
{
  Year: 2015,
  Month: 'February',
  State: 'Minnesota',
  County: 'Clearwater County',
  Rate: 12.7
```



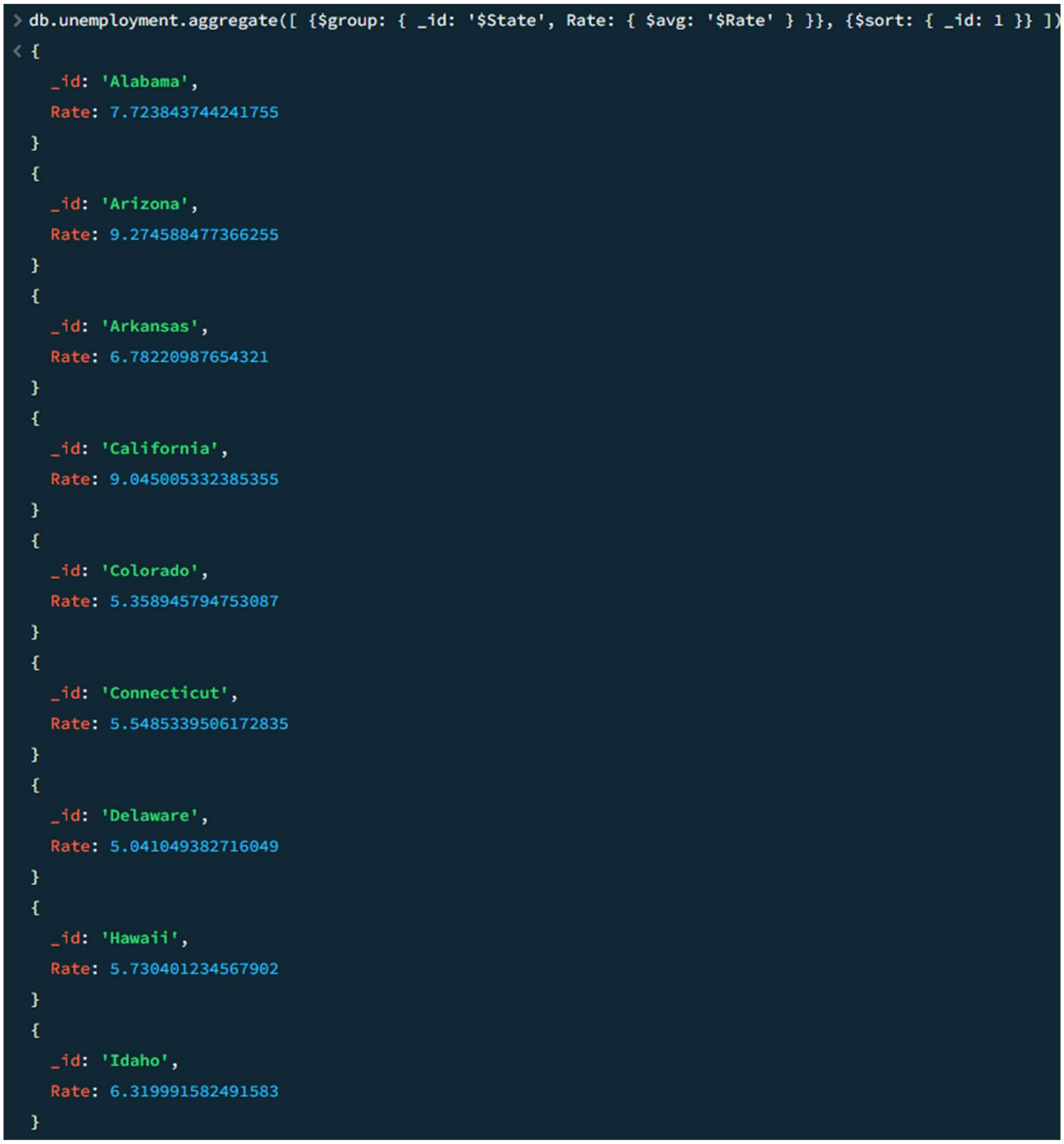
```
}
{
  Year: 2015,
  Month: 'February',
  State: 'Minnesota',
  County: 'Clearwater County',
  Rate: 12.7
}
{
  Year: 2015,
  Month: 'February',
  State: 'Arkansas',
  County: 'Jackson County',
  Rate: 10.1
}
{
  Year: 2015,
  Month: 'February',
  State: 'Arkansas',
  County: 'Chicot County',
  Rate: 11.9
}
{
  Year: 2015,
  Month: 'February',
  State: 'Arkansas',
  County: 'Mississippi County',
  Rate: 10.2
}
{
  Year: 2015,
  Month: 'February',
  State: 'New Mexico',
  County: 'Luna County',
  Rate: 21.2
}
Type "it" for more
```

Problem 5:

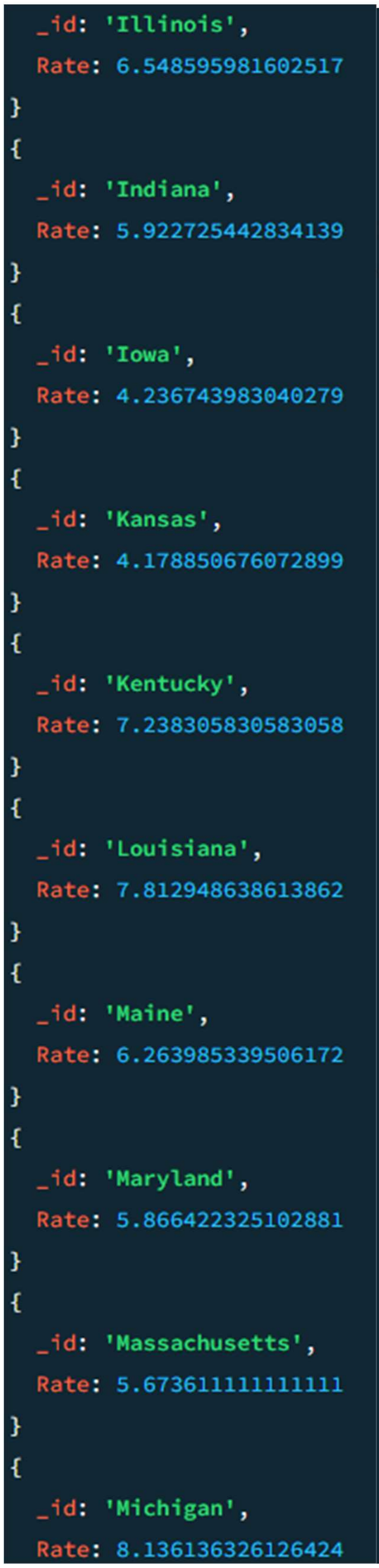
To calculate the average unemployment rate across all states, I used an aggregate query. I started with a group stage to group on the name of the state and find the average unemployment rate per state (over the entire collection period). I then used a sort stage to order the states alphabetically (in ascending order due to the “1” being utilized). I also used one assumption for the problem:

- We can assume that the question is asking for each state (individually) over the entire collection period.

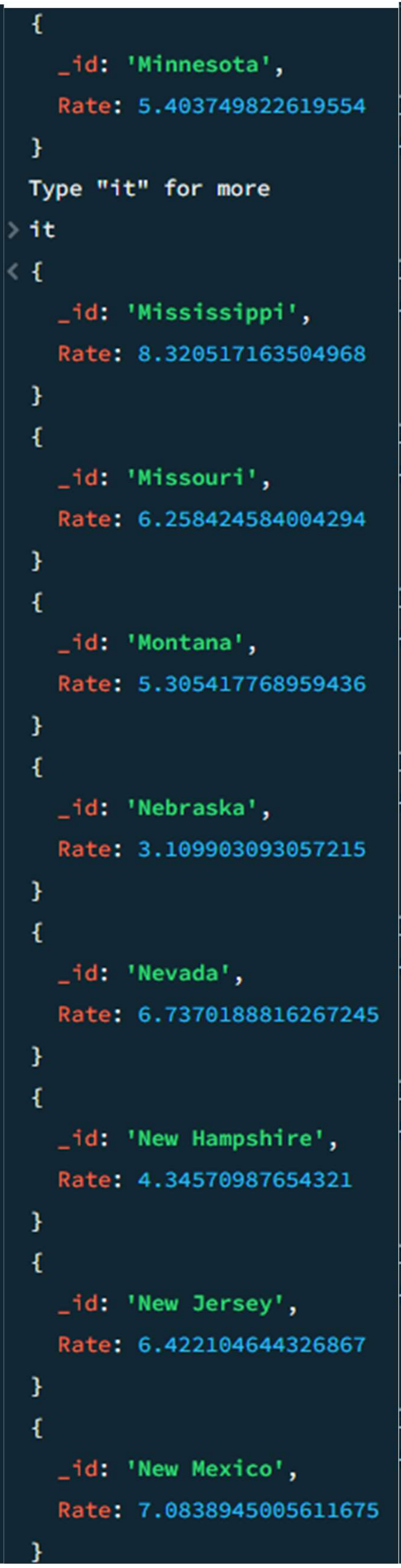
Pictures of the output are shown below.



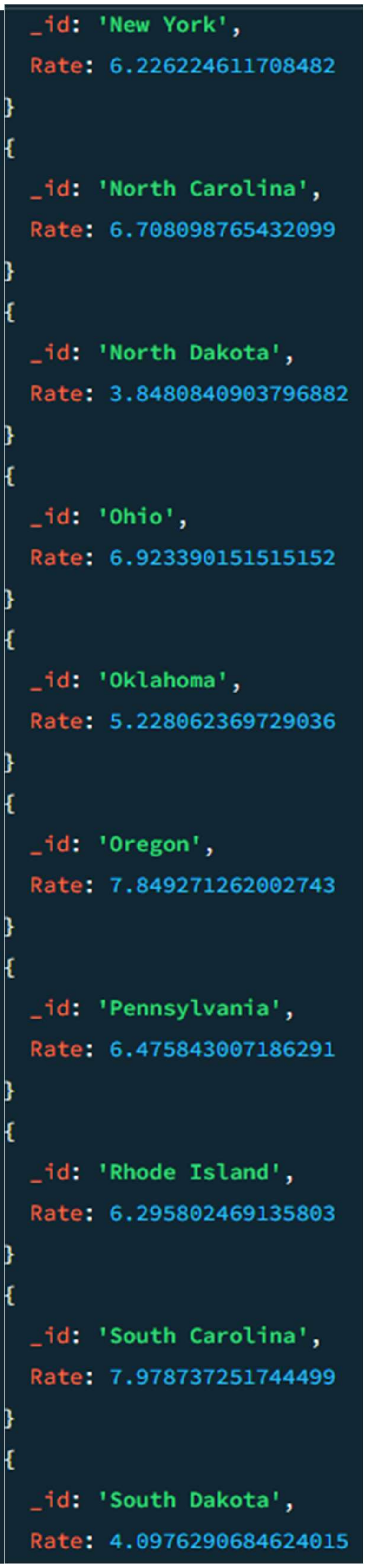
```
> db.unemployment.aggregate([ {$group: { _id: '$State', Rate: { $avg: '$Rate' } }}, {$sort: { _id: 1 }} ])
< {
    _id: 'Alabama',
    Rate: 7.723843744241755
  }
  {
    _id: 'Arizona',
    Rate: 9.274588477366255
  }
  {
    _id: 'Arkansas',
    Rate: 6.78220987654321
  }
  {
    _id: 'California',
    Rate: 9.045005332385355
  }
  {
    _id: 'Colorado',
    Rate: 5.358945794753087
  }
  {
    _id: 'Connecticut',
    Rate: 5.5485339506172835
  }
  {
    _id: 'Delaware',
    Rate: 5.041049382716049
  }
  {
    _id: 'Hawaii',
    Rate: 5.730401234567902
  }
  {
    _id: 'Idaho',
    Rate: 6.319991582491583
  }
```



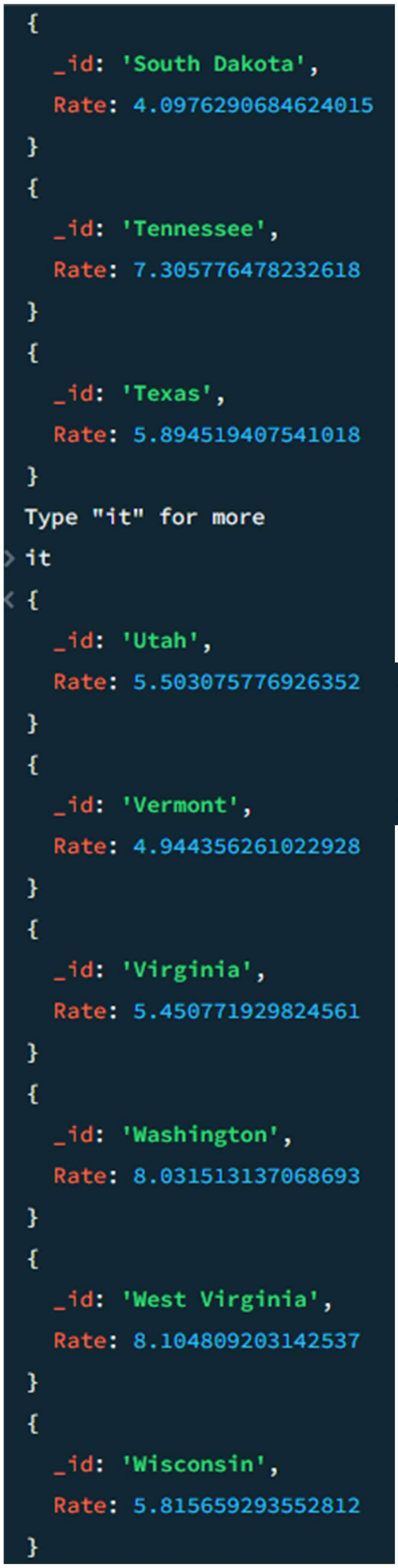
```
  _id: 'Illinois',
  Rate: 6.548595981602517
}
{
  _id: 'Indiana',
  Rate: 5.922725442834139
}
{
  _id: 'Iowa',
  Rate: 4.236743983040279
}
{
  _id: 'Kansas',
  Rate: 4.178850676072899
}
{
  _id: 'Kentucky',
  Rate: 7.238305830583058
}
{
  _id: 'Louisiana',
  Rate: 7.812948638613862
}
{
  _id: 'Maine',
  Rate: 6.263985339506172
}
{
  _id: 'Maryland',
  Rate: 5.866422325102881
}
{
  _id: 'Massachusetts',
  Rate: 5.673611111111111
}
{
  _id: 'Michigan',
  Rate: 8.136136326126424
```



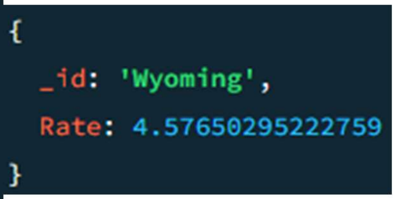
```
{
  _id: 'Minnesota',
  Rate: 5.403749822619554
}
Type "it" for more
> it
< {
  _id: 'Mississippi',
  Rate: 8.320517163504968
}
{
  _id: 'Missouri',
  Rate: 6.258424584004294
}
{
  _id: 'Montana',
  Rate: 5.305417768959436
}
{
  _id: 'Nebraska',
  Rate: 3.109903093057215
}
{
  _id: 'Nevada',
  Rate: 6.7370188816267245
}
{
  _id: 'New Hampshire',
  Rate: 4.34570987654321
}
{
  _id: 'New Jersey',
  Rate: 6.422104644326867
}
{
  _id: 'New Mexico',
  Rate: 7.0838945005611675
}
```



```
  _id: 'New York',
  Rate: 6.226224611708482
}
{
  _id: 'North Carolina',
  Rate: 6.708098765432099
}
{
  _id: 'North Dakota',
  Rate: 3.8480840903796882
}
{
  _id: 'Ohio',
  Rate: 6.923390151515152
}
{
  _id: 'Oklahoma',
  Rate: 5.228062369729036
}
{
  _id: 'Oregon',
  Rate: 7.849271262002743
}
{
  _id: 'Pennsylvania',
  Rate: 6.475843007186291
}
{
  _id: 'Rhode Island',
  Rate: 6.295802469135803
}
{
  _id: 'South Carolina',
  Rate: 7.978737251744499
}
{
  _id: 'South Dakota',
  Rate: 4.0976290684624015
```



```
{
  _id: 'South Dakota',
  Rate: 4.0976290684624015
}
{
  _id: 'Tennessee',
  Rate: 7.305776478232618
}
{
  _id: 'Texas',
  Rate: 5.894519407541018
}
Type "it" for more
> it
< {
  _id: 'Utah',
  Rate: 5.503075776926352
}
{
  _id: 'Vermont',
  Rate: 4.944356261022928
}
{
  _id: 'Virginia',
  Rate: 5.450771929824561
}
{
  _id: 'Washington',
  Rate: 8.031513137068693
}
{
  _id: 'West Virginia',
  Rate: 8.104809203142537
}
{
  _id: 'Wisconsin',
  Rate: 5.815659293552812
}
```



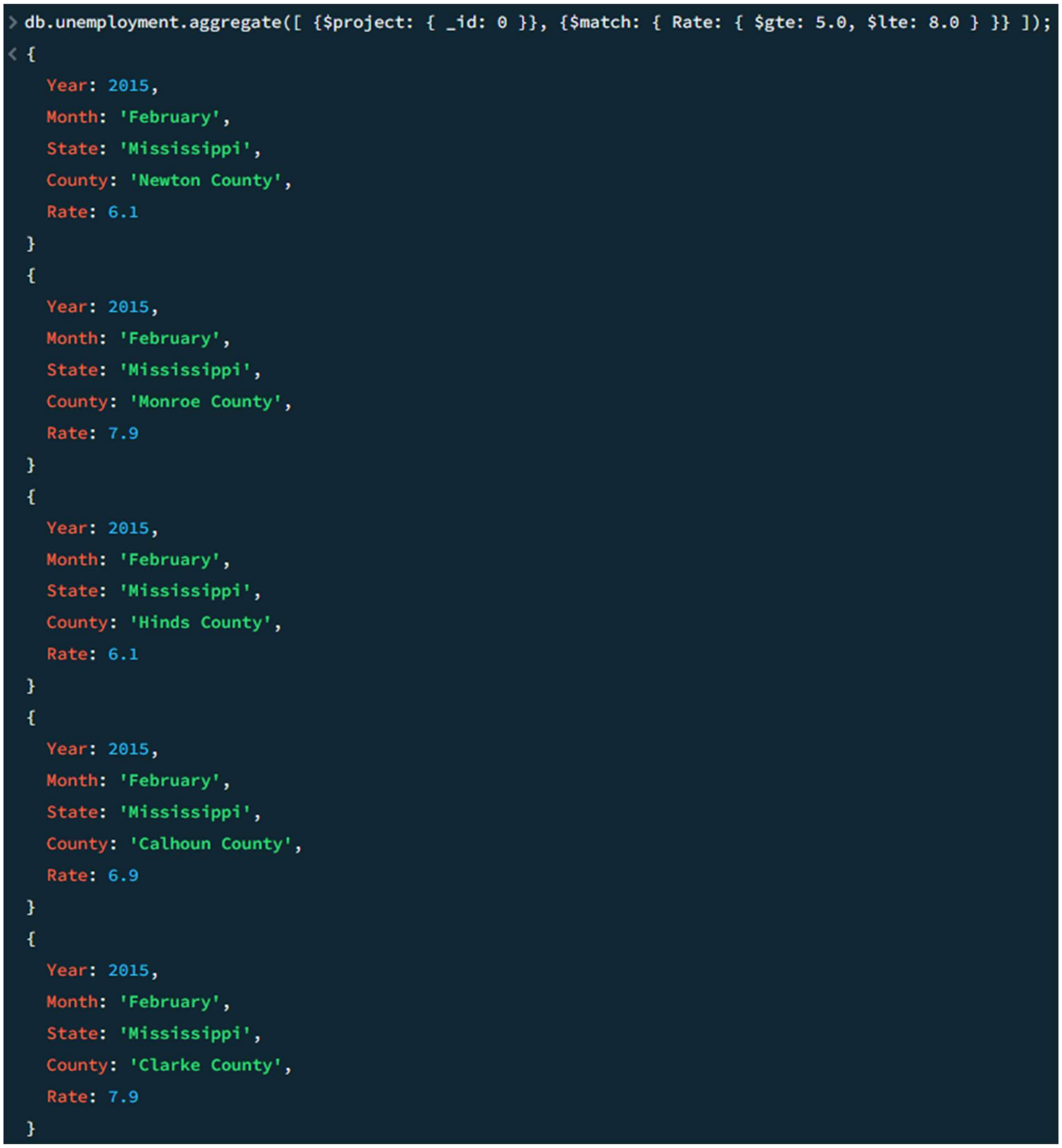
```
{
  _id: 'Wyoming',
  Rate: 4.57650295222759
}
```

Problem 6:

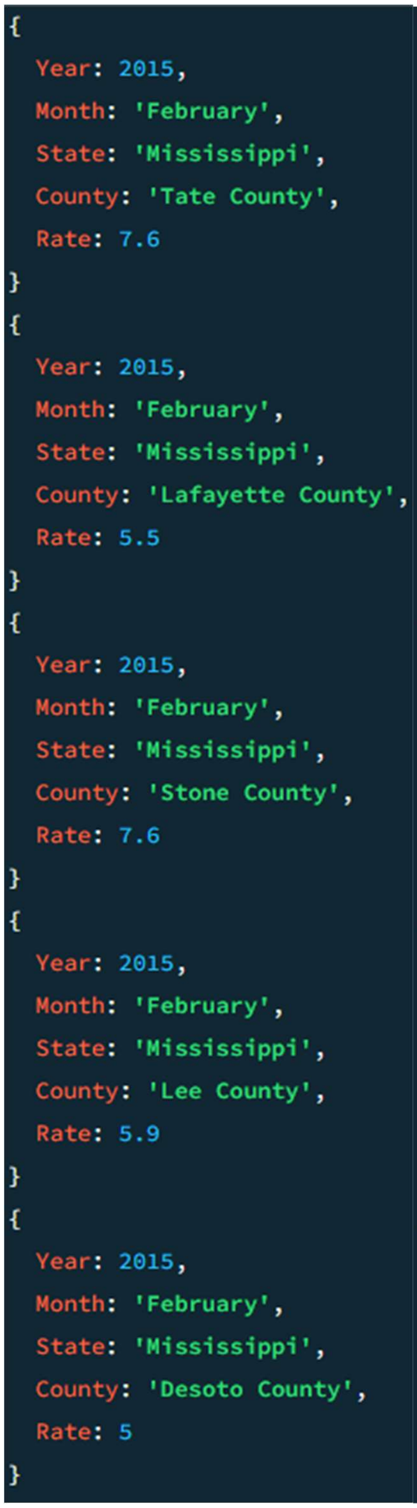
To find all counties with an unemployment rate between 5% and 8%, I created an aggregate query like that of problem 4’s query. I started with a project stage to get rid of the “_id” field for the output where I then used a match stage to find all the documents where the unemployment rate is between 5.0% and 8.0%, thus producing the counties with their respective state, month, year, and rate (all the relevant data). I also used three assumptions for the problem:

- We can assume that the question is asking for the rate to be [5.0, 8.0] (inclusive).
- We can assume that the question is asking monthly per each state's county (with a rate of 5.0-8.0%) and NOT an average all time OR per year.
- We can assume that the question is asking for the output to be a list of counties (including relevant data) and NOT the size of the list.

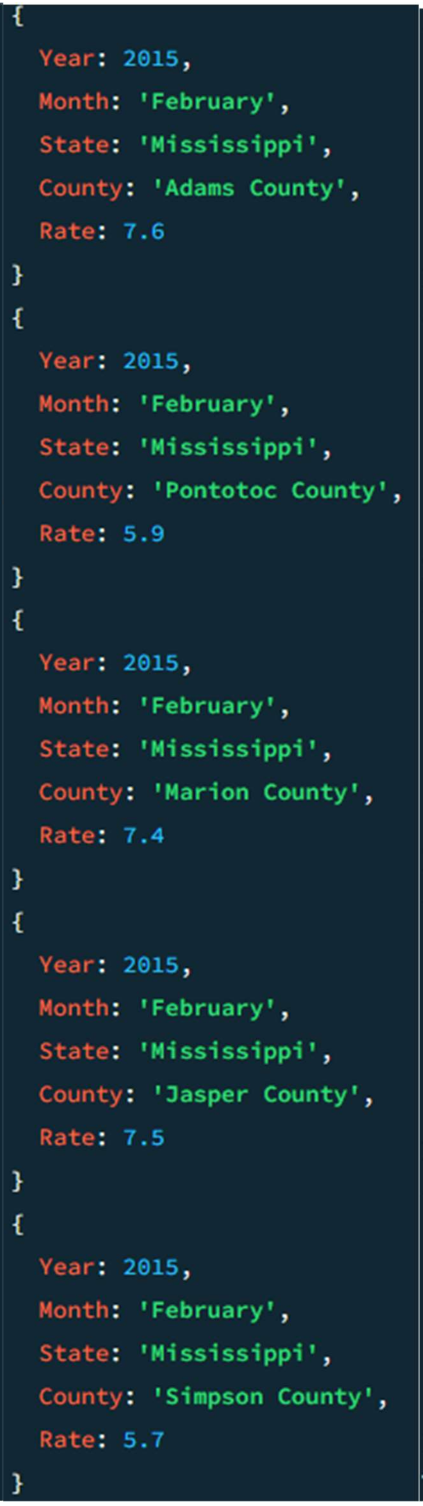
Pictures of the output are shown below.



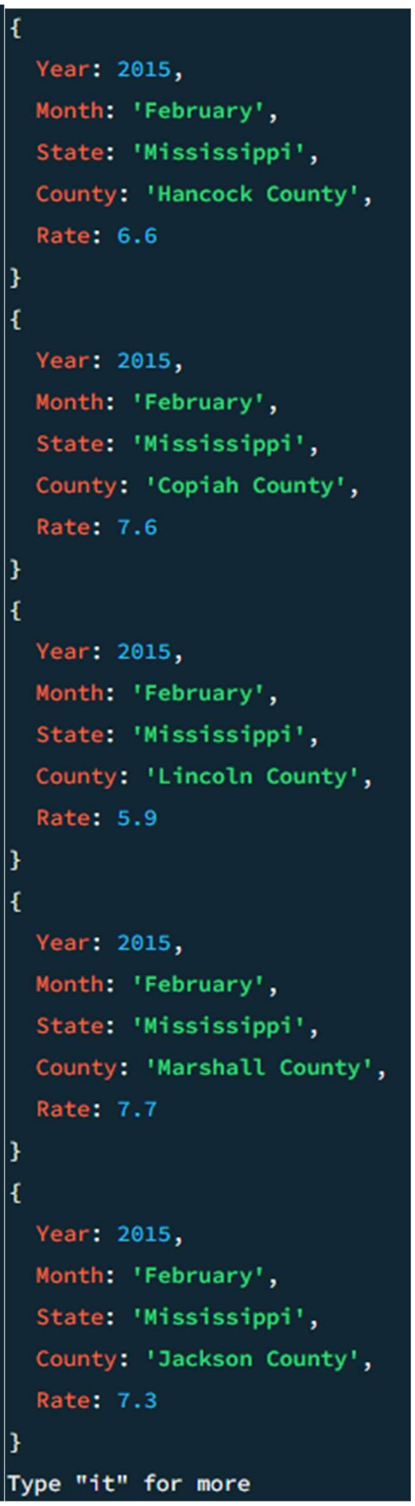
```
> db.unemployment.aggregate([ {$project: { _id: 0 }}, {$match: { Rate: { $gte: 5.0, $lte: 8.0 } }} ]);
< {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Newton County',
    Rate: 6.1
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Monroe County',
    Rate: 7.9
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Hinds County',
    Rate: 6.1
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Calhoun County',
    Rate: 6.9
  }
  {
    Year: 2015,
    Month: 'February',
    State: 'Mississippi',
    County: 'Clarke County',
    Rate: 7.9
  }
```



```
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Tate County',
  Rate: 7.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Lafayette County',
  Rate: 5.5
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Stone County',
  Rate: 7.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Lee County',
  Rate: 5.9
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Desoto County',
  Rate: 5
}
```



```
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Adams County',
  Rate: 7.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Pontotoc County',
  Rate: 5.9
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Marion County',
  Rate: 7.4
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Jasper County',
  Rate: 7.5
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Simpson County',
  Rate: 5.7
}
```

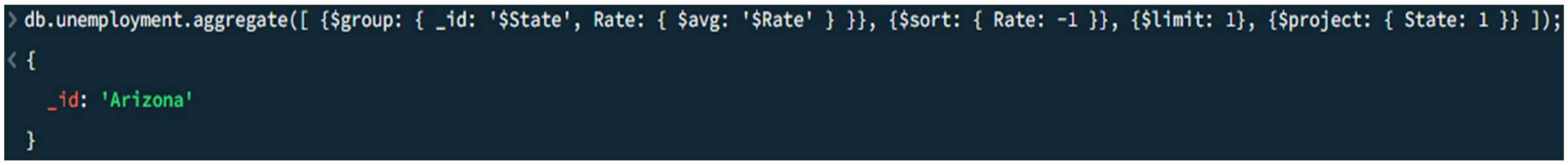


```
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Hancock County',
  Rate: 6.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Copiah County',
  Rate: 7.6
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Lincoln County',
  Rate: 5.9
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Marshall County',
  Rate: 7.7
}
{
  Year: 2015,
  Month: 'February',
  State: 'Mississippi',
  County: 'Jackson County',
  Rate: 7.3
}
Type "it" for more
```

Problem 7:

An aggregate query was created to find the state with the highest unemployment rate. I started with a group stage where I group on the state and find the average rate per each state over the entire collection period. I then use a sort stage to sort the rates in descending order where I then use a limit stage to grab the top value, finally ending with a project to only show the name of the state in the output (Arizona had the highest average unemployment rate over the entire collection). I also used one assumption for the problem:

- We can assume that the question is asking for the state with the highest average unemployment rate over the entire collection period

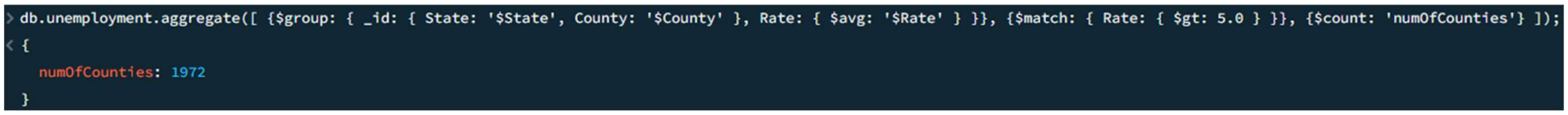


```
> db.unemployment.aggregate([ {$group: { _id: '$State', Rate: { $avg: '$Rate' } }}, {$sort: { Rate: -1 }}, {$limit: 1}, {$project: { State: 1 }} ]);
< {
    _id: 'Arizona'
  }
```

Problem 8:

Similarly to problem 6's query, I used an aggregate query to count how many counties have an unemployment rate above 5%. I started with a group stage where I grouped on the state and the state's specific county where I then found the average rate for the state's county over then entire collection period. I then used a match stage to only return the average rates that are greater than 5.0% and finally ended with a count stage to return the "numOfCounties". I also used two assumptions for the problem:

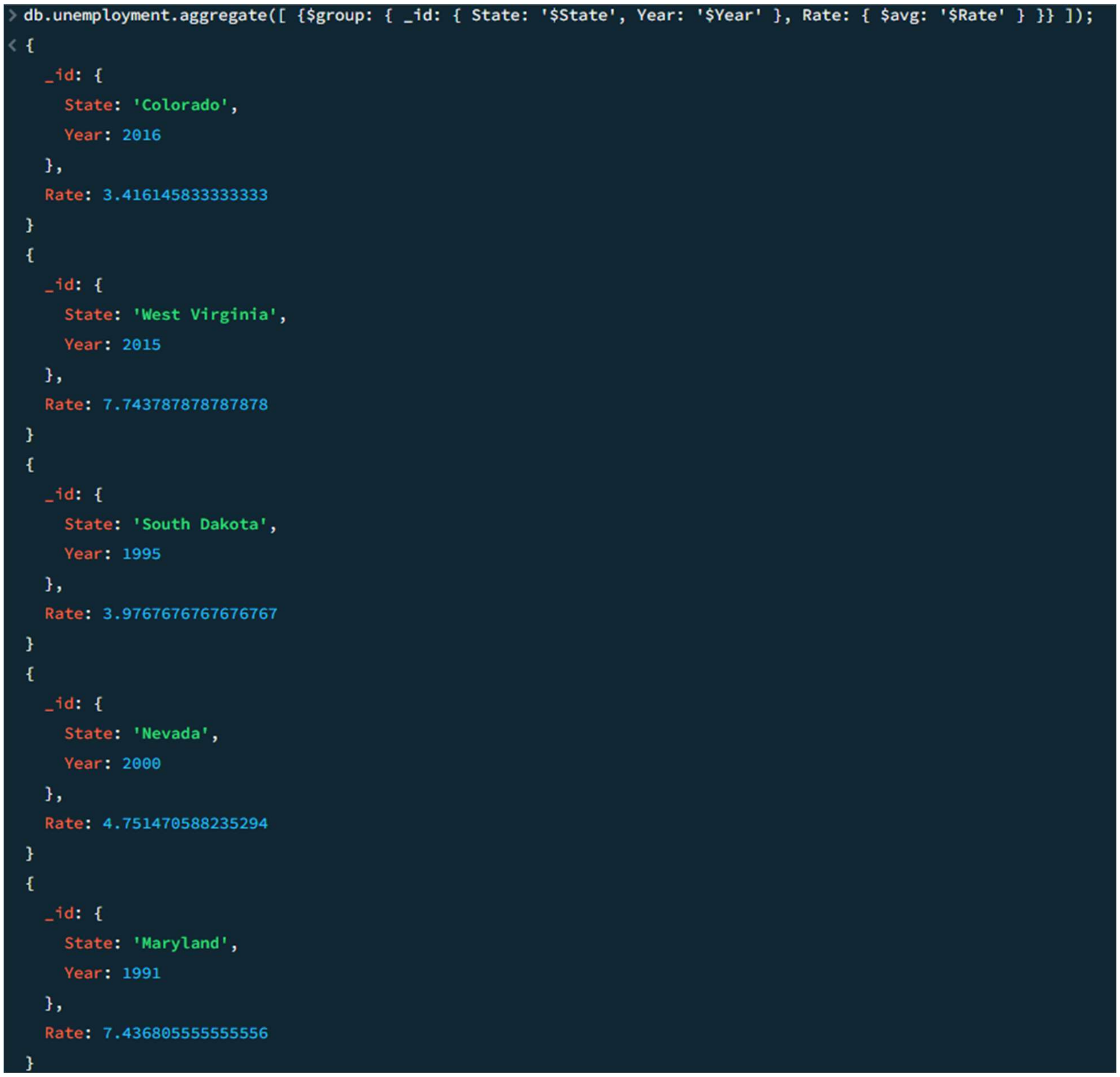
- We can assume that the question is asking for each state's county's average rate over the entire collection period.
- We can assume that the question is asking for the rate to be (5.0, infinity) (exclusive).



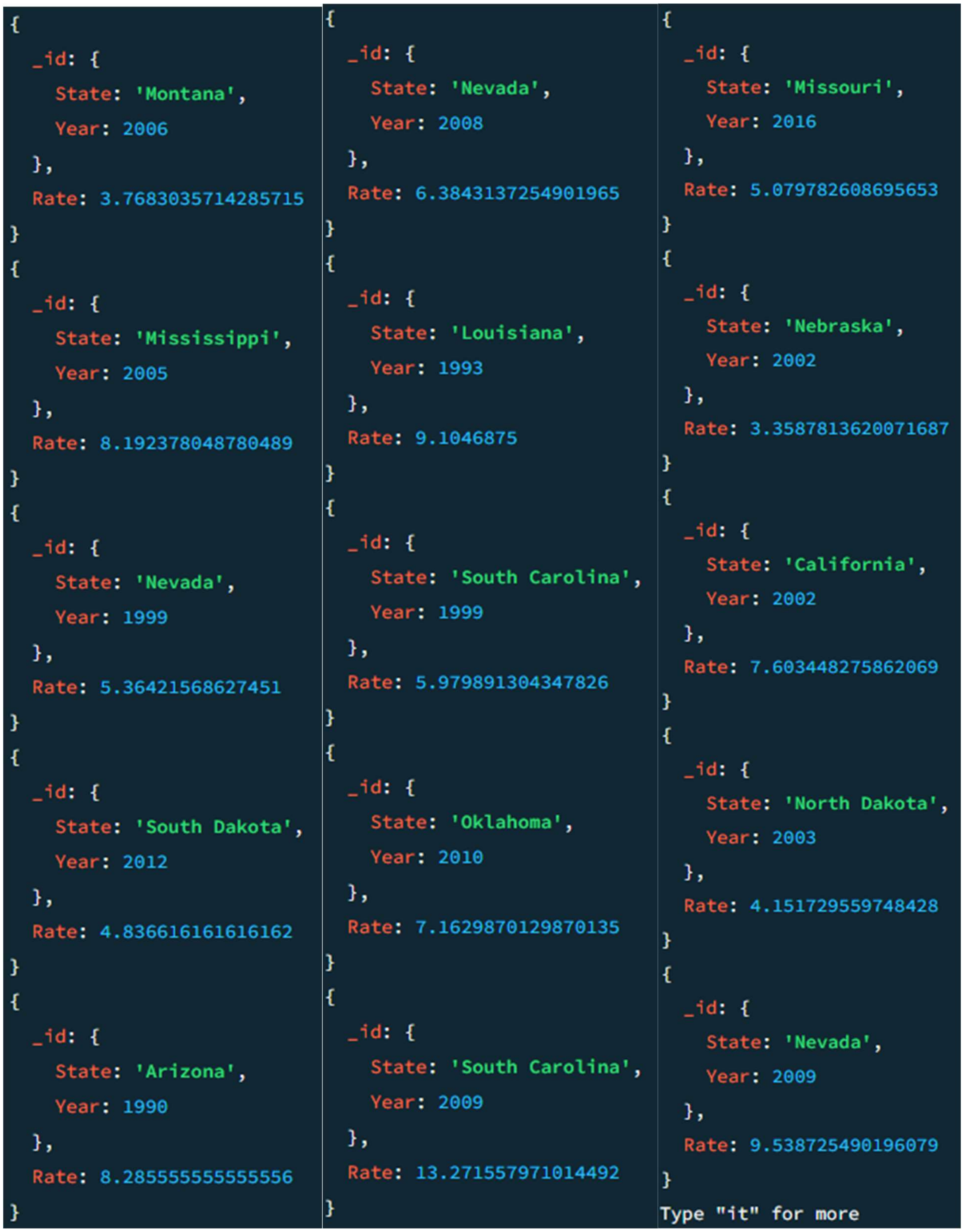
```
> db.unemployment.aggregate([ {$group: { _id: { State: '$State', County: '$County' }, Rate: { $avg: '$Rate' } }}, {$match: { Rate: { $gt: 5.0 } }}, {$count: 'numOfCounties'} ]);
< {
    numOfCounties: 1972
  }
```

Problem 9:

To calculate the average unemployment rate per state by year, I created an aggregate query where I made a group stage to group on the state and the year, following with the calculation of the average unemployment rate.



```
> db.unemployment.aggregate([ {$group: { _id: { State: '$State', Year: '$Year' }, Rate: { $avg: '$Rate' } }} ]);
< {
    _id: {
      State: 'Colorado',
      Year: 2016
    },
    Rate: 3.416145833333333
  }
  {
    _id: {
      State: 'West Virginia',
      Year: 2015
    },
    Rate: 7.743787878787878
  }
  {
    _id: {
      State: 'South Dakota',
      Year: 1995
    },
    Rate: 3.9767676767676767
  }
  {
    _id: {
      State: 'Nevada',
      Year: 2000
    },
    Rate: 4.751470588235294
  }
  {
    _id: {
      State: 'Maryland',
      Year: 1991
    },
    Rate: 7.436805555555556
  }
```

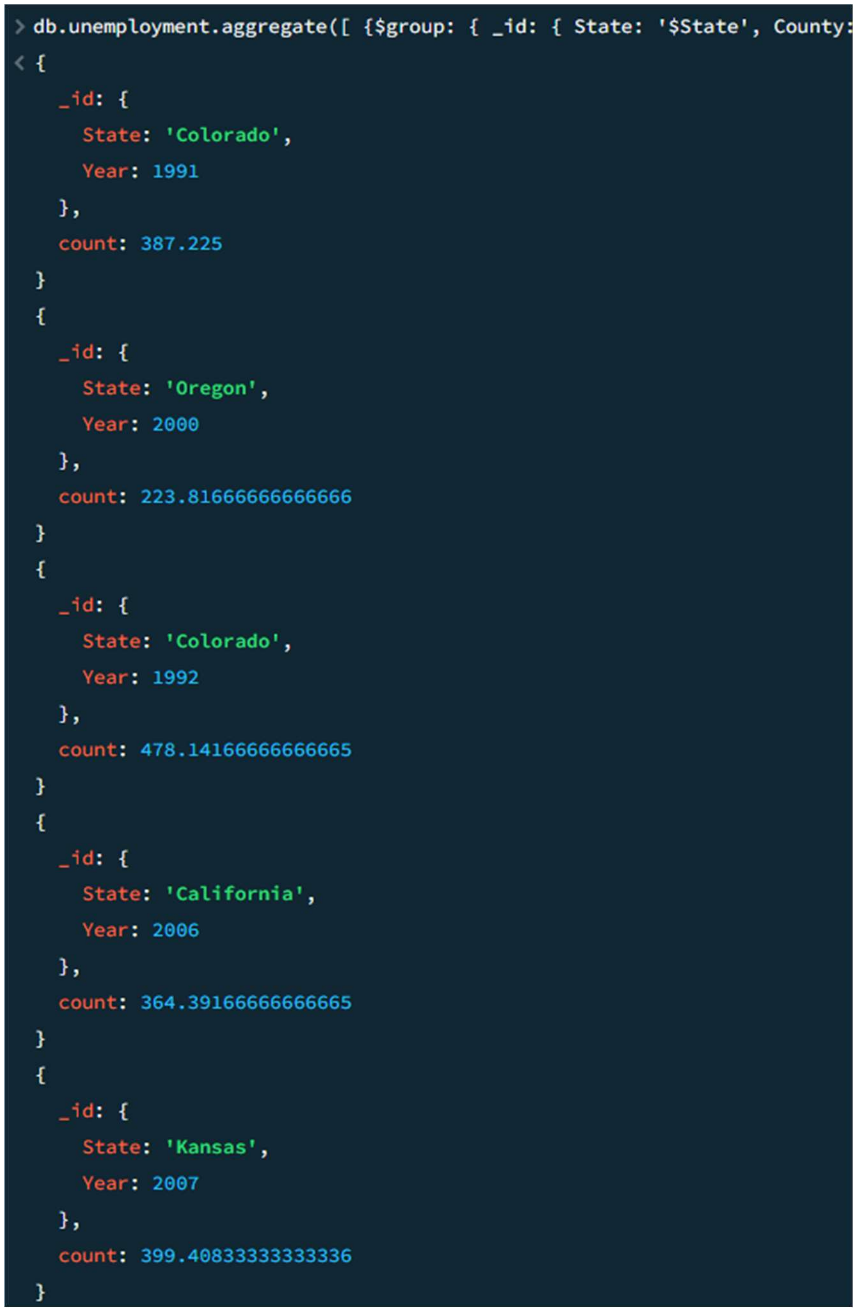


```
{
  _id: {
    State: 'Montana',
    Year: 2006
  },
  Rate: 3.7683035714285715
}
{
  _id: {
    State: 'Mississippi',
    Year: 2005
  },
  Rate: 8.192378048780489
}
{
  _id: {
    State: 'Nevada',
    Year: 1999
  },
  Rate: 5.36421568627451
}
{
  _id: {
    State: 'South Dakota',
    Year: 2012
  },
  Rate: 4.836616161616162
}
{
  _id: {
    State: 'Arizona',
    Year: 1990
  },
  Rate: 8.285555555555556
}
{
  _id: {
    State: 'Nevada',
    Year: 2008
  },
  Rate: 6.3843137254901965
}
{
  _id: {
    State: 'Louisiana',
    Year: 1993
  },
  Rate: 9.1046875
}
{
  _id: {
    State: 'South Carolina',
    Year: 1999
  },
  Rate: 5.979891304347826
}
{
  _id: {
    State: 'Oklahoma',
    Year: 2010
  },
  Rate: 7.1629870129870135
}
{
  _id: {
    State: 'South Carolina',
    Year: 2009
  },
  Rate: 13.271557971014492
}
{
  _id: {
    State: 'Missouri',
    Year: 2016
  },
  Rate: 5.079782608695653
}
{
  _id: {
    State: 'Nebraska',
    Year: 2002
  },
  Rate: 3.3587813620071687
}
{
  _id: {
    State: 'California',
    Year: 2002
  },
  Rate: 7.603448275862069
}
{
  _id: {
    State: 'North Dakota',
    Year: 2003
  },
  Rate: 4.151729559748428
}
{
  _id: {
    State: 'Nevada',
    Year: 2009
  },
  Rate: 9.538725490196079
}
Type "it" for more
```

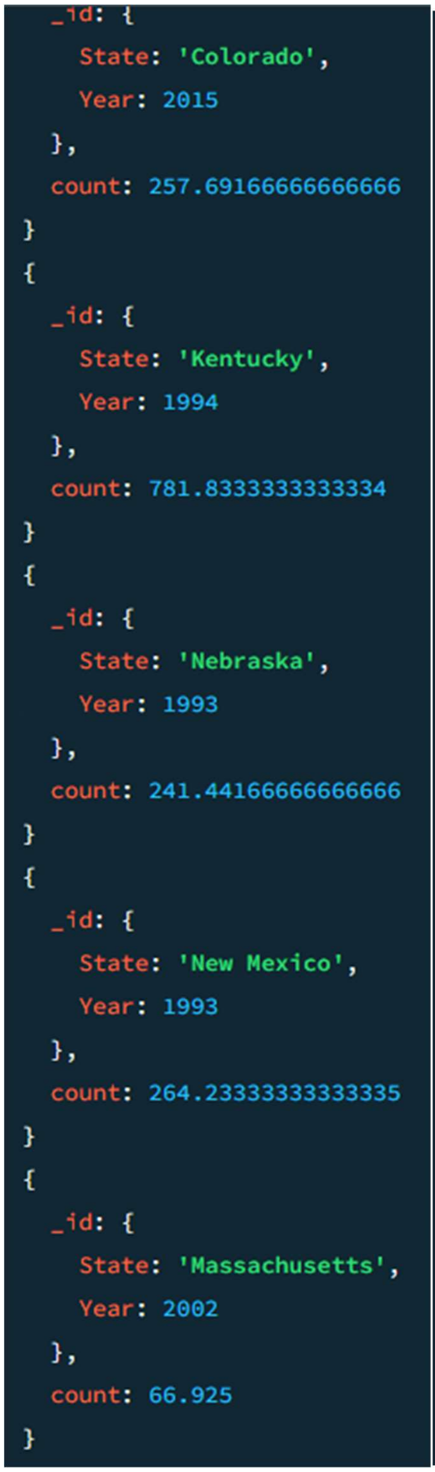
Extra Credit 1:

An aggregate query was created to calculate the total unemployment rate across all counties, for each state. I started with a group stage where I group on the state, the county, and the year to calculate the state's county's average rate in a year. I then utilize another group stage to group on the fields within the objects that were just created, causing the use of dot-notation to access the state and the year where the sum of the average county rate in a year is calculated. I also used one assumption for the problem:

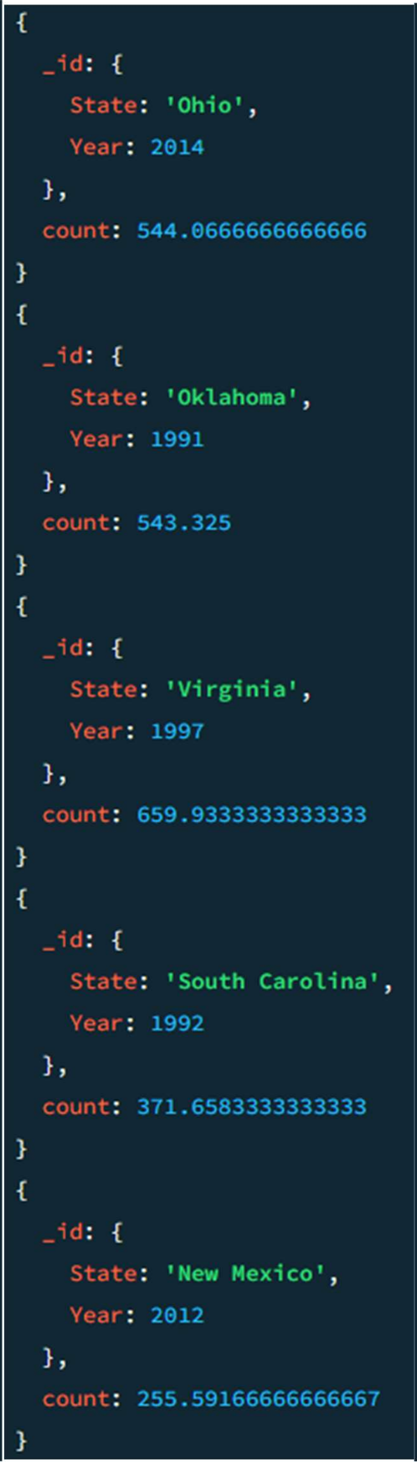
- We can assume that the question is inferring that the total rate for the states should be summed up per year.



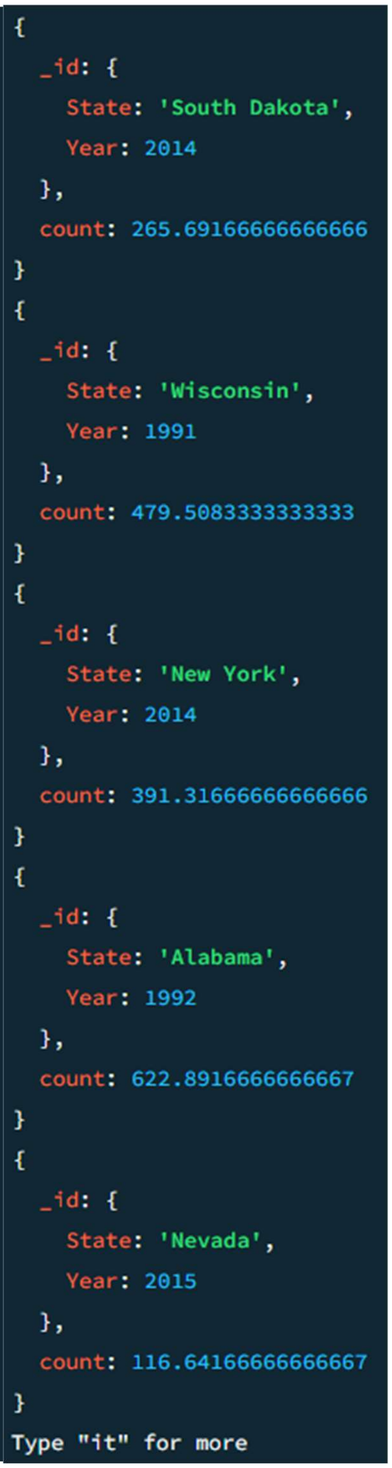
```
> db.unemployment.aggregate([ {$group: { _id: { State: '$State', County:
< {
    _id: {
      State: 'Colorado',
      Year: 1991
    },
    count: 387.225
  }
  {
    _id: {
      State: 'Oregon',
      Year: 2000
    },
    count: 223.81666666666666
  }
  {
    _id: {
      State: 'Colorado',
      Year: 1992
    },
    count: 478.14166666666665
  }
  {
    _id: {
      State: 'California',
      Year: 2006
    },
    count: 364.39166666666665
  }
  {
    _id: {
      State: 'Kansas',
      Year: 2007
    },
    count: 399.40833333333336
  }
```



```
  _id: {
    State: 'Colorado',
    Year: 2015
  },
  count: 257.69166666666666
}
{
  _id: {
    State: 'Kentucky',
    Year: 1994
  },
  count: 781.8333333333334
}
{
  _id: {
    State: 'Nebraska',
    Year: 1993
  },
  count: 241.44166666666666
}
{
  _id: {
    State: 'New Mexico',
    Year: 1993
  },
  count: 264.23333333333335
}
{
  _id: {
    State: 'Massachusetts',
    Year: 2002
  },
  count: 66.925
}
```



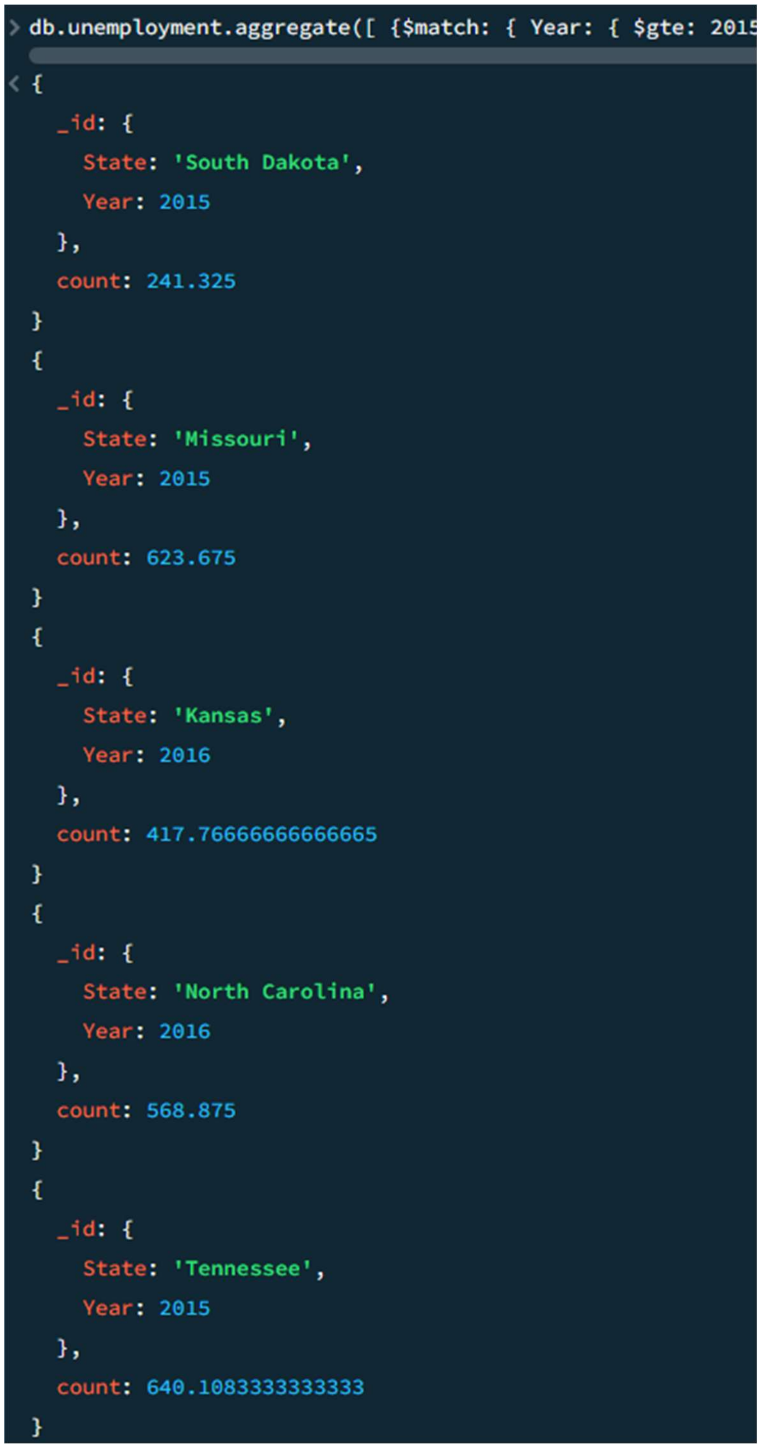
```
{
  _id: {
    State: 'Ohio',
    Year: 2014
  },
  count: 544.0666666666666
}
{
  _id: {
    State: 'Oklahoma',
    Year: 1991
  },
  count: 543.325
}
{
  _id: {
    State: 'Virginia',
    Year: 1997
  },
  count: 659.9333333333333
}
{
  _id: {
    State: 'South Carolina',
    Year: 1992
  },
  count: 371.6583333333333
}
{
  _id: {
    State: 'New Mexico',
    Year: 2012
  },
  count: 255.59166666666667
}
```



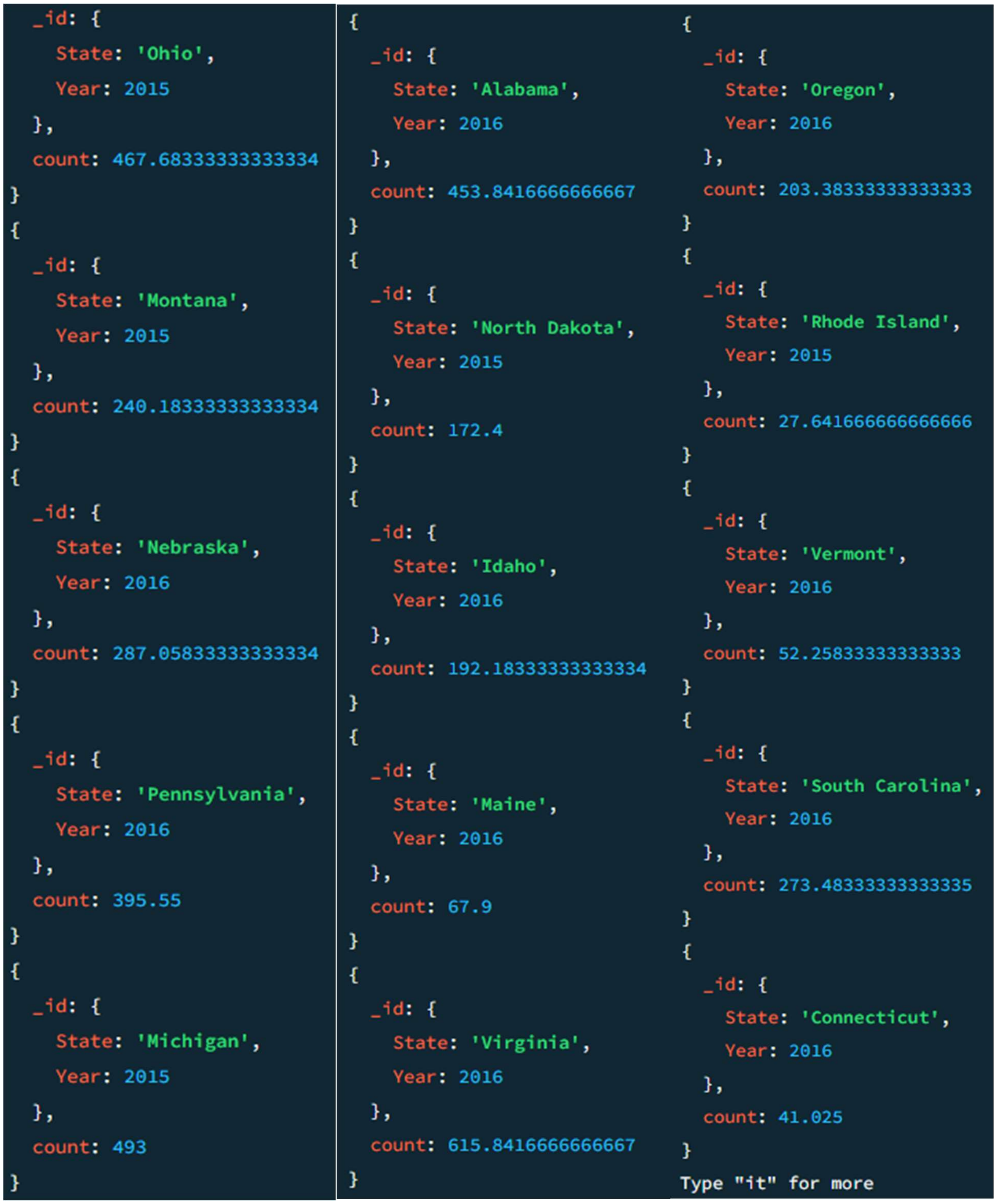
```
{
  _id: {
    State: 'South Dakota',
    Year: 2014
  },
  count: 265.69166666666666
}
{
  _id: {
    State: 'Wisconsin',
    Year: 1991
  },
  count: 479.5083333333333
}
{
  _id: {
    State: 'New York',
    Year: 2014
  },
  count: 391.31666666666666
}
{
  _id: {
    State: 'Alabama',
    Year: 1992
  },
  count: 622.8916666666667
}
{
  _id: {
    State: 'Nevada',
    Year: 2015
  },
  count: 116.64166666666667
}
Type "it" for more
```

Extra Credit 2:

Building off the first extra credit problem query but with just 2015 and on data being produced the same steps were taken from the previous problem but with a match stage added as the first stage in the aggregate query. The match stage was used as the starting filter to reduce the set of documents before running the 2 group stages on them; here, the filter finds all the documents that are from 2015 and on with the greater than or equal to query.



```
> db.unemployment.aggregate([ {$match: { Year: { $gte: 2015
< {
    _id: {
      State: 'South Dakota',
      Year: 2015
    },
    count: 241.325
  }
  {
    _id: {
      State: 'Missouri',
      Year: 2015
    },
    count: 623.675
  }
  {
    _id: {
      State: 'Kansas',
      Year: 2016
    },
    count: 417.76666666666665
  }
  {
    _id: {
      State: 'North Carolina',
      Year: 2016
    },
    count: 568.875
  }
  {
    _id: {
      State: 'Tennessee',
      Year: 2015
    },
    count: 640.1083333333333
  }
```



```
  _id: {
    State: 'Ohio',
    Year: 2015
  },
  count: 467.68333333333334
}
{
  _id: {
    State: 'Montana',
    Year: 2015
  },
  count: 240.18333333333334
}
{
  _id: {
    State: 'Nebraska',
    Year: 2016
  },
  count: 287.05833333333334
}
{
  _id: {
    State: 'Pennsylvania',
    Year: 2016
  },
  count: 395.55
}
{
  _id: {
    State: 'Michigan',
    Year: 2015
  },
  count: 493
}
{
  _id: {
    State: 'Alabama',
    Year: 2016
  },
  count: 453.8416666666667
}
{
  _id: {
    State: 'North Dakota',
    Year: 2015
  },
  count: 172.4
}
{
  _id: {
    State: 'Idaho',
    Year: 2016
  },
  count: 192.18333333333334
}
{
  _id: {
    State: 'Maine',
    Year: 2016
  },
  count: 67.9
}
{
  _id: {
    State: 'Virginia',
    Year: 2016
  },
  count: 615.8416666666667
}
{
  _id: {
    State: 'Oregon',
    Year: 2016
  },
  count: 203.38333333333333
}
{
  _id: {
    State: 'Rhode Island',
    Year: 2015
  },
  count: 27.641666666666666
}
{
  _id: {
    State: 'Vermont',
    Year: 2016
  },
  count: 52.25833333333333
}
{
  _id: {
    State: 'South Carolina',
    Year: 2016
  },
  count: 273.48333333333335
}
{
  _id: {
    State: 'Connecticut',
    Year: 2016
  },
  count: 41.025
}
Type "it" for more
```