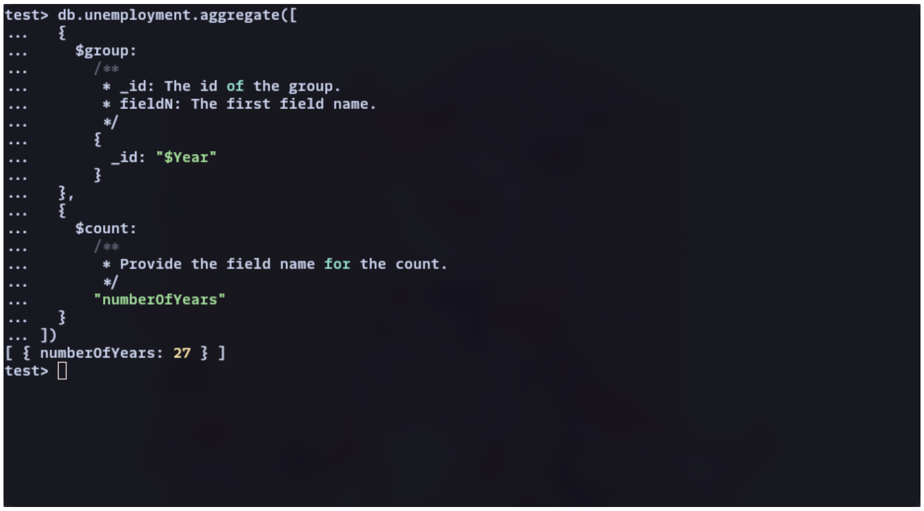
Question 1:

Over how many years was the unemployment data collected?

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        /**
         * Gather years
         */
        {
            _id: "$Year"
        }
    },
    {
        $count:
            /**
             * Count years
             */
            "numberOfYears"
    }
])
```

Image



```
test> 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.
...        */
...       "numberOfYears"
...   }
... ])
[ { numberOfYears: 27 } ]
test> 
```

Explanation

The query groups the years together and gets the count of the unique years in that group.

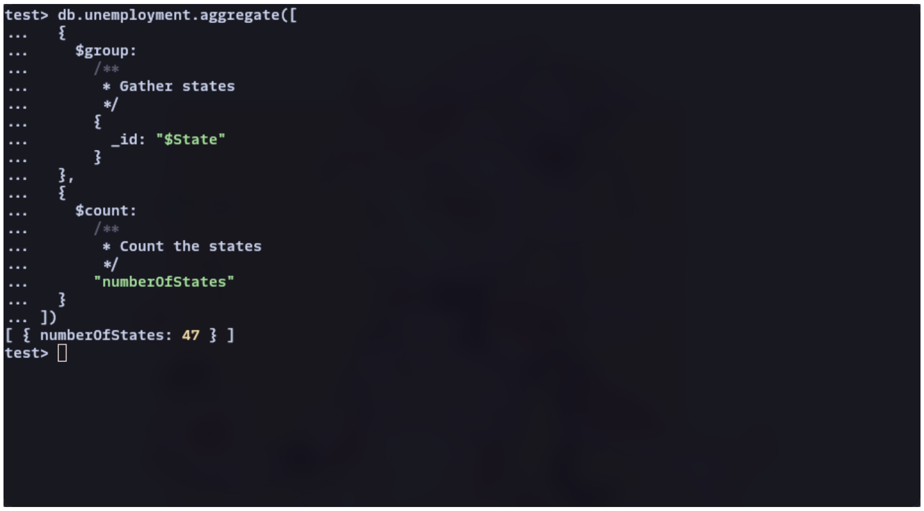
Question 2:

How many states were reported on in this dataset?

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        /**
         * Gather states
         */
        {
            _id: "$State"
        }
    },
    {
        $count:
            /**
             * Count the states
             */
            "numberOfStates"
    }
])
```

Image



```
test> db.unemployment.aggregate([
...   {
...     $group:
...       /**
...        * Gather states
...        */
...       {
...         _id: "$State"
...       }
...   },
...   {
...     $count:
...       /**
...        * Count the states
...        */
...       "numberOfStates"
...   }
... ])
[ { numberOfStates: 47 } ]
test> 
```

Explanation

The query groups the states together and counts the unique states similar to problem 1.

Question 3:

What does this query compute?

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

Explanation

That query finds documents which have a unemployment rate which is less than 1.0 then counts the amount of documents found.

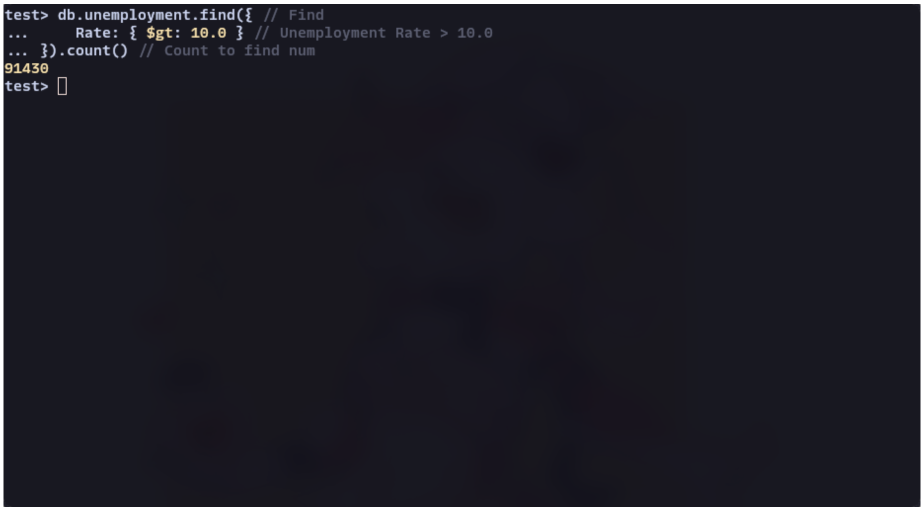
Question 4:

Find all counties with unemployment rate higher than 10%

JS Code

```
db.unemployment.find({ // Find
    Rate: { $gt: 10.0 } // Unemployment Rate > 10.0
}).count() // Count to find num
```

Image



```
test> db.unemployment.find({ // Find
...     Rate: { $gt: 10.0 } // Unemployment Rate > 10.0
... }).count() // Count to find num
91430
test> 
```

Explanation

The query finds documents with unemployment rates above 10.0 then counts the number found.

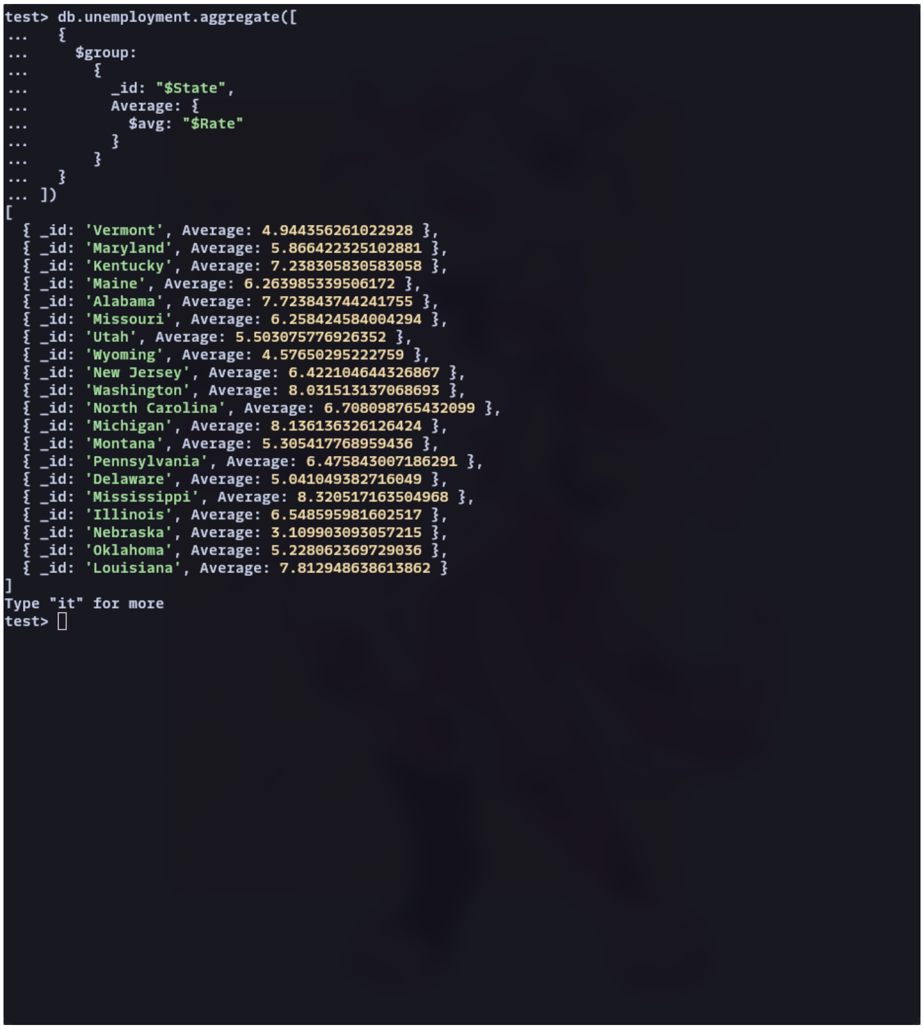
Question 5:

Calculate the average unemployment rate across all states.

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        {
            _id: "$State", // Group by state
            Average: { // Find average rate
                $avg: "$Rate"
            }
        }
    }
])
```

Image



```
test> db.unemployment.aggregate([
...   {
...     $group:
...       {
...         _id: "$State",
...         Average: {
...           $avg: "$Rate"
...         }
...       }
...   }
... ])
[
  { _id: 'Vermont', Average: 4.944356261022928 },
  { _id: 'Maryland', Average: 5.866422325102881 },
  { _id: 'Kentucky', Average: 7.238305830583058 },
  { _id: 'Maine', Average: 6.263985339506172 },
  { _id: 'Alabama', Average: 7.723843744241755 },
  { _id: 'Missouri', Average: 6.258424584004294 },
  { _id: 'Utah', Average: 5.503075776926352 },
  { _id: 'Wyoming', Average: 4.57650295222759 },
  { _id: 'New Jersey', Average: 6.422104644326867 },
  { _id: 'Washington', Average: 8.031513137068693 },
  { _id: 'North Carolina', Average: 6.708098765432099 },
  { _id: 'Michigan', Average: 8.136136326126424 },
  { _id: 'Montana', Average: 5.305417768959436 },
  { _id: 'Pennsylvania', Average: 6.475843007186291 },
  { _id: 'Delaware', Average: 5.041049382716049 },
  { _id: 'Mississippi', Average: 8.320517163504968 },
  { _id: 'Illinois', Average: 6.548595981602517 },
  { _id: 'Nebraska', Average: 3.109903093057215 },
  { _id: 'Oklahoma', Average: 5.228062369729036 },
  { _id: 'Louisiana', Average: 7.812948638613862 }
]
Type "it" for more
test> 
```

Explanation

The query groups the states together and computes the average unemployment rate for the gathered states.

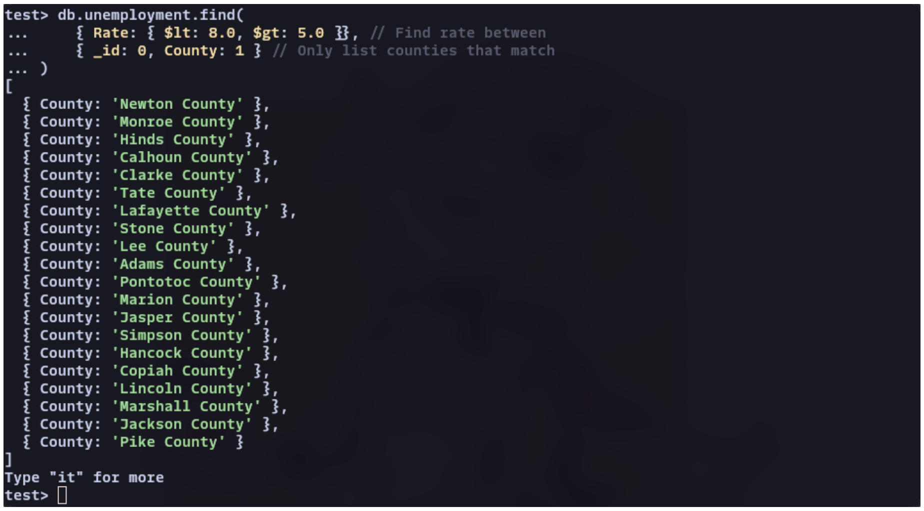
Question 6:

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

JS Code

```
db.unemployment.find(
    { Rate: { $lt: 8.0, $gt: 5.0 } }, // Find rate between
    { _id: 0, County: 1 } // Only list counties that match
)
```

Image



```
test> db.unemployment.find(
...     { Rate: { $lt: 8.0, $gt: 5.0 }}, // Find rate between
...     { _id: 0, County: 1 } // Only list counties that match
... )
[
  { County: 'Newton County' },
  { County: 'Monroe County' },
  { County: 'Hinds County' },
  { County: 'Calhoun County' },
  { County: 'Clarke County' },
  { County: 'Tate County' },
  { County: 'Lafayette County' },
  { County: 'Stone County' },
  { County: 'Lee County' },
  { County: 'Adams County' },
  { County: 'Pontotoc County' },
  { County: 'Marion County' },
  { County: 'Jasper County' },
  { County: 'Simpson County' },
  { County: 'Hancock County' },
  { County: 'Copiah County' },
  { County: 'Lincoln County' },
  { County: 'Marshall County' },
  { County: 'Jackson County' },
  { County: 'Pike County' }
]
Type "it" for more
test> 
```

Explanation

The query finds documents with an unemployment rate greater than 5.0 and less than 8.0 the selects only the County element to output from the matching documents.

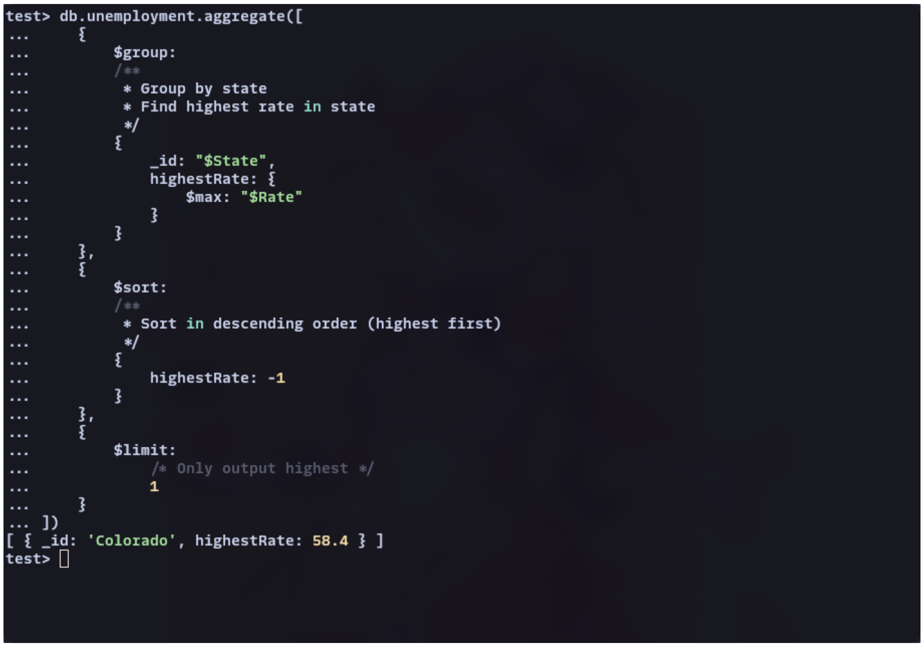
Question 7:

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

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        /**
         * Group by state
         * Find highest rate in state
         */
        {
            _id: "$State",
            highestRate: {
                $max: "$Rate"
            }
        }
    },
    {
        $sort:
        /**
         * Sort in descending order (highest first)
         */
        {
            highestRate: -1
        }
    },
    {
        $limit:
            /* Only output highest */
            1
    }
])
```

Image



```
test> db.unemployment.aggregate([
...     {
...         $group:
...         /**
...          * Group by state
...          * Find highest rate in state
...          */
...         {
...             _id: "$State",
...             highestRate: {
...                 $max: "$Rate"
...             }
...         }
...     },
...     {
...         $sort:
...         /**
...          * Sort in descending order (highest first)
...          */
...         {
...             highestRate: -1
...         }
...     },
...     {
...         $limit:
...             /* Only output highest */
...             1
...     }
... ])
[ { _id: 'Colorado', highestRate: 58.4 } ]
test> 
```

Explanation

The query groups by state and searches for the max unemployment rate in the gathered states. It then sorts those rates in descending order and limits to the first output which will be the largest unemployment rate.

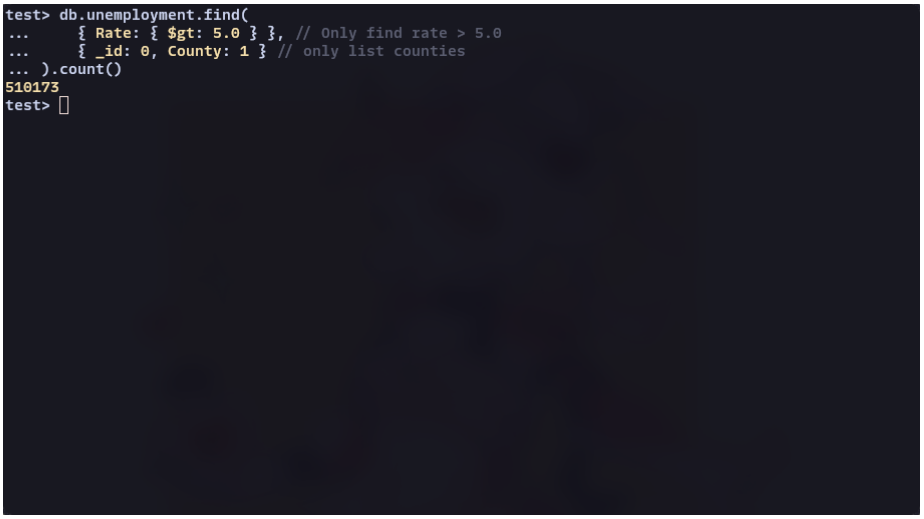
Question 8:

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

JS Code

```
db.unemployment.find(
    { Rate: { $gt: 5.0 } }, // Only find rate > 5.0
    { _id: 0, County: 1 } // only list counties
).count()
```

Image



```
test> db.unemployment.find(
...     { Rate: { $gt: 5.0 } }, // Only find rate > 5.0
...     { _id: 0, County: 1 } // only list counties
... ).count()
510173
test> 
```

Explanation

The query finds documents with an unemployment rate above 5.0 and only outputs the counties. Then it counts the amount output to find the totaly number.

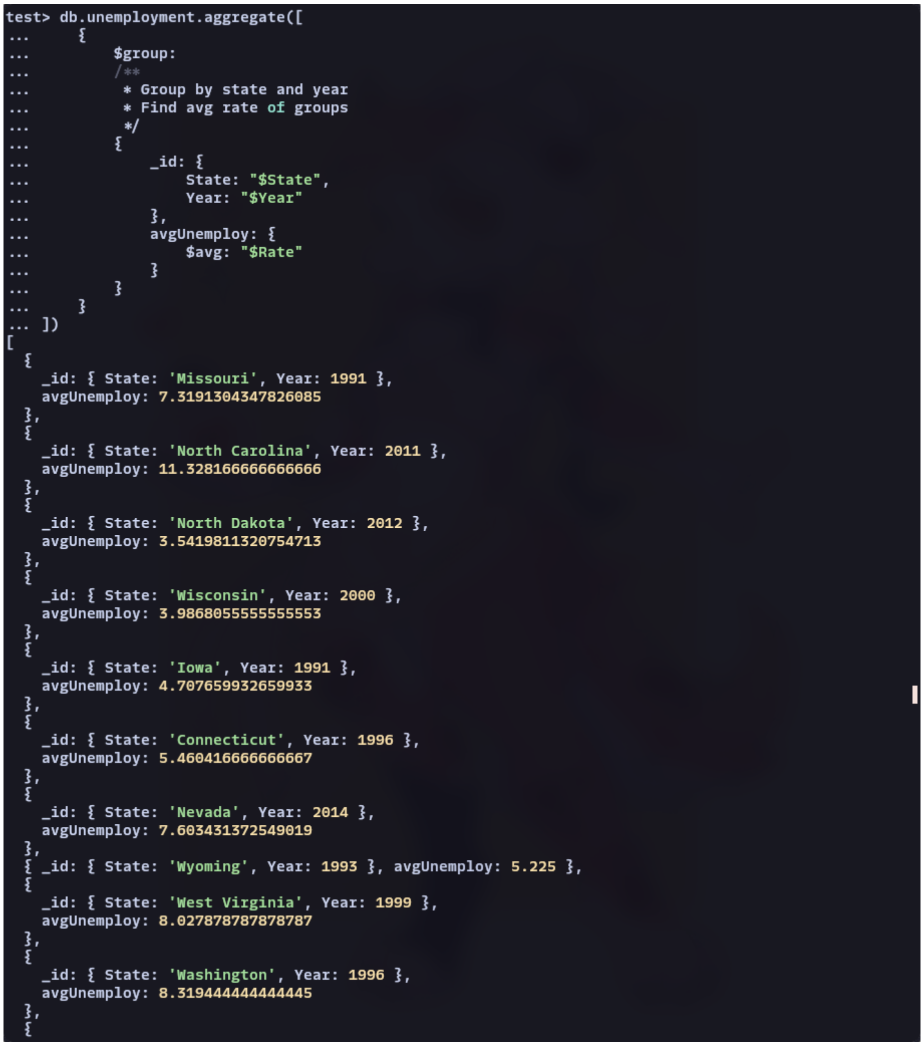
Question 9:

Calculate the average unemployment rate per state by year.

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        /**
         * Group by state and year
         * Find avg rate of groups
         */
        {
            _id: {
                State: "$State",
                Year: "$Year"
            },
            avgUnemploy: {
                $avg: "$Rate"
            }
        }
    }
])
```

Image



```
test> db.unemployment.aggregate([
...     {
...         $group:
...         /**
...          * Group by state and year
...          * Find avg rate of groups
...          */
...         {
...             _id: {
...                 State: "$State",
...                 Year: "$Year"
...             },
...             avgUnemploy: {
...                 $avg: "$Rate"
...             }
...         }
...     }
... ])
[
  {
    _id: { State: 'Missouri', Year: 1991 },
    avgUnemploy: 7.3191304347826085
  },
  {
    _id: { State: 'North Carolina', Year: 2011 },
    avgUnemploy: 11.328166666666666
  },
  {
    _id: { State: 'North Dakota', Year: 2012 },
    avgUnemploy: 3.5419811320754713
  },
  {
    _id: { State: 'Wisconsin', Year: 2000 },
    avgUnemploy: 3.9868055555555553
  },
  {
    _id: { State: 'Iowa', Year: 1991 },
    avgUnemploy: 4.707659932659933
  },
  {
    _id: { State: 'Connecticut', Year: 1996 },
    avgUnemploy: 5.460416666666667
  },
  {
    _id: { State: 'Nevada', Year: 2014 },
    avgUnemploy: 7.603431372549019
  },
  { _id: { State: 'Wyoming', Year: 1993 }, avgUnemploy: 5.225 },
  {
    _id: { State: 'West Virginia', Year: 1999 },
    avgUnemploy: 8.027878787878787
  },
  {
    _id: { State: 'Washington', Year: 1996 },
    avgUnemploy: 8.319444444444445
  },
  {
```

Explanation

The query groups by state and year. It then finds the average for unemployment rate for each of the groups.

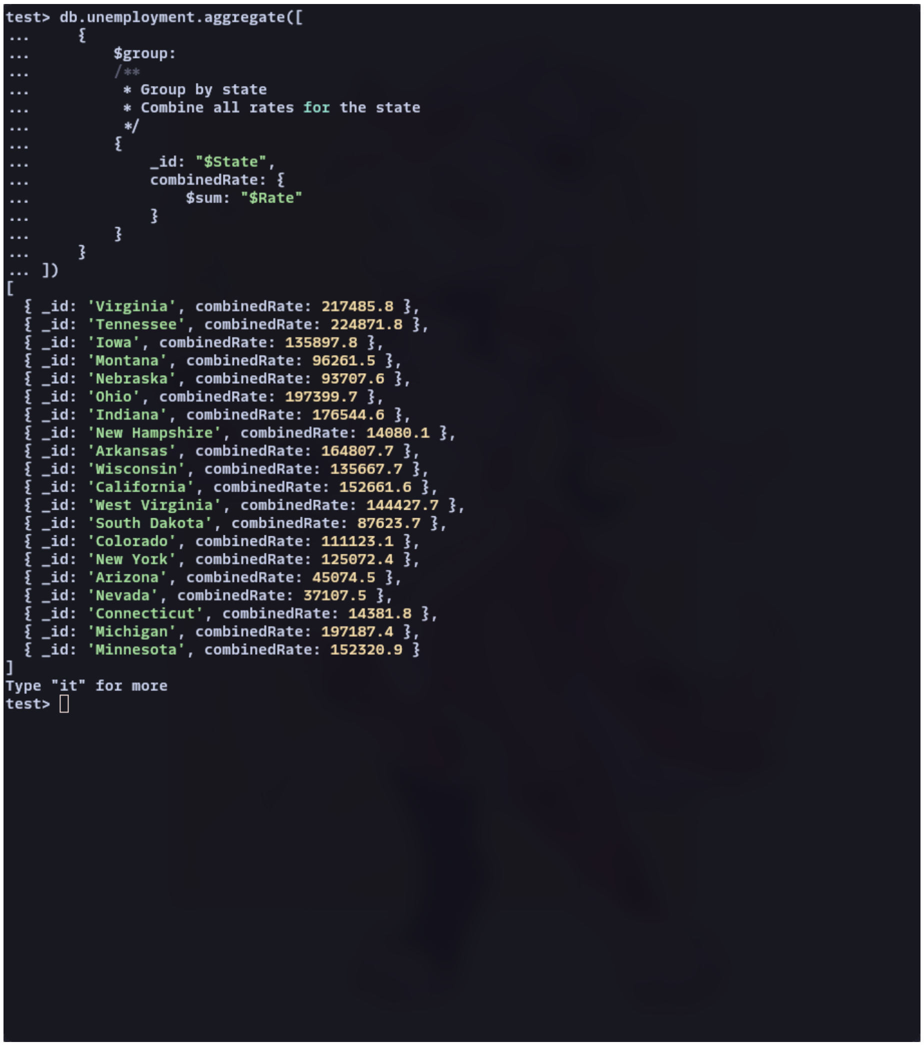
Question 10:

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

JS Code

```
db.unemployment.aggregate([
    {
        $group:
        /**
         * Group by state
         * Combine all rates for the state
         */
        {
            _id: "$State",
            combinedRate: {
                $sum: "$Rate"
            }
        }
    }
])
```

Image



```
test> db.unemployment.aggregate([
...     {
...         $group:
...         /**
...          * Group by state
...          * Combine all rates for the state
...          */
...         {
...             _id: "$State",
...             combinedRate: {
...                 $sum: "$Rate"
...             }
...         }
...     }
... ])
[
  { _id: 'Virginia', combinedRate: 217485.8 },
  { _id: 'Tennessee', combinedRate: 224871.8 },
  { _id: 'Iowa', combinedRate: 135897.8 },
  { _id: 'Montana', combinedRate: 96261.5 },
  { _id: 'Nebraska', combinedRate: 93707.6 },
  { _id: 'Ohio', combinedRate: 197399.7 },
  { _id: 'Indiana', combinedRate: 176544.6 },
  { _id: 'New Hampshire', combinedRate: 14080.1 },
  { _id: 'Arkansas', combinedRate: 164807.7 },
  { _id: 'Wisconsin', combinedRate: 135667.7 },
  { _id: 'California', combinedRate: 152661.6 },
  { _id: 'West Virginia', combinedRate: 144427.7 },
  { _id: 'South Dakota', combinedRate: 87623.7 },
  { _id: 'Colorado', combinedRate: 111123.1 },
  { _id: 'New York', combinedRate: 125072.4 },
  { _id: 'Arizona', combinedRate: 45074.5 },
  { _id: 'Nevada', combinedRate: 37107.5 },
  { _id: 'Connecticut', combinedRate: 14381.8 },
  { _id: 'Michigan', combinedRate: 197187.4 },
  { _id: 'Minnesota', combinedRate: 152320.9 }
]
Type "it" for more
test> 
```

Explanation

The query groups documents by state then finds the combined unemployment rate for all of the documents in the group- each being a county.

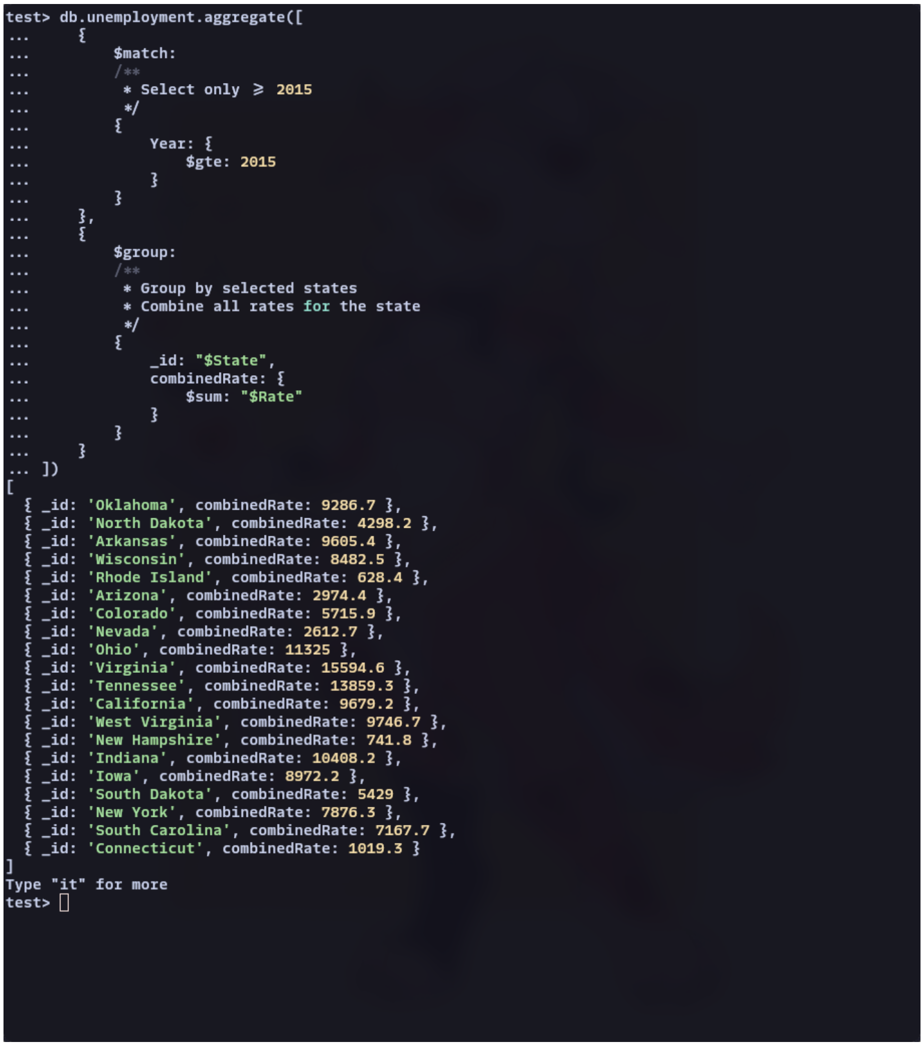
Question 11:

The same as Query 10 but for states with data from 2015 onward

JS Code

```
db.unemployment.aggregate([
    {
        $match:
        /**
         * Select only >= 2015
         */
        {
            Year: {
                $gte: 2015
            }
        }
    },
    {
        $group:
        /**
         * Group by selected states
         * Combine all rates for the state
         */
        {
            _id: "$State",
            combinedRate: {
                $sum: "$Rate"
            }
        }
    }
])
```

Image



```
test> db.unemployment.aggregate([
...     {
...         $match:
...         /**
...          * Select only ⩾ 2015
...          */
...         {
...             Year: {
...                 $gte: 2015
...             }
...         }
...     },
...     {
...         $group:
...         /**
...          * Group by selected states
...          * Combine all rates for the state
...          */
...         {
...             _id: "$State",
...             combinedRate: {
...                 $sum: "$Rate"
...             }
...         }
...     }
... ])
[
  { _id: 'Oklahoma', combinedRate: 9286.7 },
  { _id: 'North Dakota', combinedRate: 4298.2 },
  { _id: 'Arkansas', combinedRate: 9605.4 },
  { _id: 'Wisconsin', combinedRate: 8482.5 },
  { _id: 'Rhode Island', combinedRate: 628.4 },
  { _id: 'Arizona', combinedRate: 2974.4 },
  { _id: 'Colorado', combinedRate: 5715.9 },
  { _id: 'Nevada', combinedRate: 2612.7 },
  { _id: 'Ohio', combinedRate: 11325 },
  { _id: 'Virginia', combinedRate: 15594.6 },
  { _id: 'Tennessee', combinedRate: 13859.3 },
  { _id: 'California', combinedRate: 9679.2 },
  { _id: 'West Virginia', combinedRate: 9746.7 },
  { _id: 'New Hampshire', combinedRate: 741.8 },
  { _id: 'Indiana', combinedRate: 10408.2 },
  { _id: 'Iowa', combinedRate: 8972.2 },
  { _id: 'South Dakota', combinedRate: 5429 },
  { _id: 'New York', combinedRate: 7876.3 },
  { _id: 'South Carolina', combinedRate: 7167.7 },
  { _id: 'Connecticut', combinedRate: 1019.3 }
]
Type "it" for more
test> ▯
```

Explanation

this query does the same as 10 but add a match at the beggining to only allow documents with years past 2015 to be aggregated.