DB Assignment 5

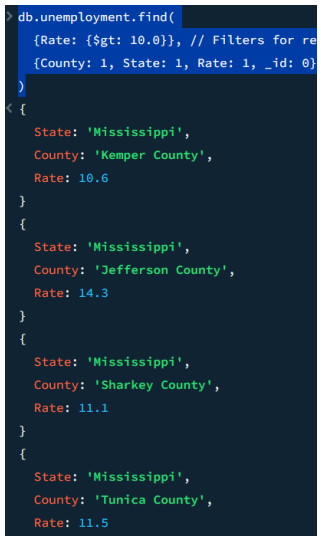
Brennan Duff

November 20, 2024

1. Over how many years was the unemployment data collected?
    a. Find the total number of distinct years.
    b. Query:
        i. db.unemployment.distinct("Year").length
    c. Screenshot:
        i.
```
> db.unemployment.distinct("Year").length
< 27
```
    d. Explanation:
        i. The distinct function retrieves all unique values for the Year field. The .length at the end counts the number of unique years, giving the total number of years for which unemployment data is available.

2. How many states were reported on in this dataset?
    a. Find the total number of distinct states.
    b. Query:
        i. db.unemployment.distinct("State").length
    c. Screenshot:
        i.
```
> db.unemployment.distinct("State").length
< 47
```
    d. Explanation:
        i. Same as first query but with the State field instead of Year.

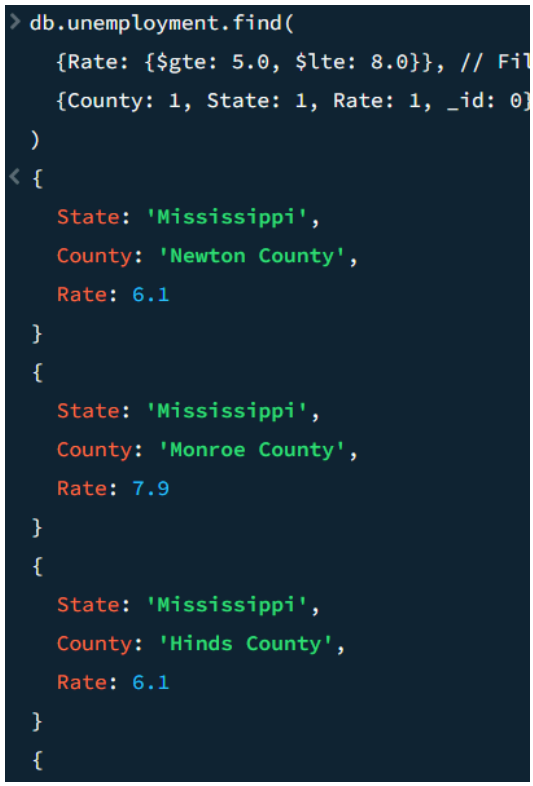
3. What does this query compute?
    a. Run the query and find what it computes.
    b. Query:
        i. db.unemployment.find({Rate : {$lt: 1.0}}).count()
    c. Screenshot:
        i.
```
> db.unemployment.find({Rate: {$lt: 1.0}}).count()
< 657
```
    d. Explanation:
        i. This query retrieves all records where the Rate field (unemployment rate) is less than 1.0. The .count() function then returns the total number of such records.

4. Find all counties with unemployment rate higher than 10%.
    a. Find the total number of counties with a rate greater than 10.
    b. Query:
        i. db.unemployment.find(
        ii. {Rate: {$gt: 10.0}},
        iii. {County: 1, State: 1, Rate: 1, _id: 0}
        iv.)
    c. Screenshot:
        i. 
```
> db.unemployment.find(
    {Rate: {$gt: 10.0}}, // Filters for re
    {County: 1, State: 1, Rate: 1, _id: 0}
  )
< {
    State: 'Mississippi',
    County: 'Kemper County',
    Rate: 10.6
  }
  {
    State: 'Mississippi',
    County: 'Jefferson County',
    Rate: 14.3
  }
  {
    State: 'Mississippi',
    County: 'Sharkey County',
    Rate: 11.1
  }
  {
    State: 'Mississippi',
    County: 'Tunica County',
    Rate: 11.5
```
(too many results to screenshot all of them)
    d. Explanation:
        i. The find method filters records where the unemployment rate exceeds 10% (Rate: {$gt: 10.0}). The second argument specifies what is shown.

5. Calculate the average unemployment rate across all states.
    a. Group the records and take the average of Rate.
    b. Query:
        i. db.unemployment.aggregate([
        ii. {
        iii. $group: {
        iv. _id: null,
        v. averageRate: {$avg: "$Rate"}
        vi. }
        vii. }
        viii.])
    c. Screenshot:
        i.

```
> db.unemployment.aggregate([
    {
      $group: {
        _id: null, // Group all records into a single bucket
        averageRate: {$avg: "$Rate"} // Calculate the average of the Rate field
      }
    }
  ])
< {
    _id: null,
    averageRate: 6.1750097115006755
  }
```

    d. Explanation:
        i. This aggregation groups all records together using _id: null and calculates the average unemployment rate ($avg: "$Rate"). The result is a single document with the average rate across the entire dataset.

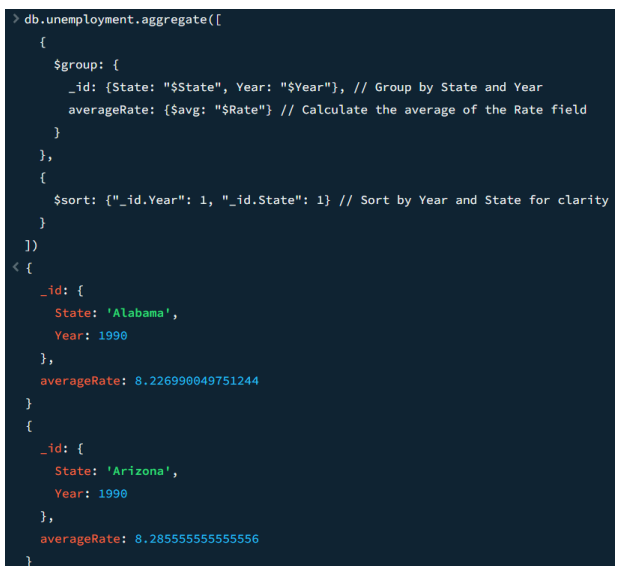
6. Find all counties with an unemployment rate between 5% and 8%.
    a. Find counties with where the Rate is less than 8 but greater than 5.
    b. Query:
        i. db.unemployment.find(
        ii. {Rate: {$gte: 5.0, $lte: 8.0}},
        iii. {County: 1, State: 1, Rate: 1, _id: 0}
        iv.)
    c. Screenshot:
        i. 
```
> db.unemployment.find(
    {Rate: {$gte: 5.0, $lte: 8.0}}, // Fil
    {County: 1, State: 1, Rate: 1, _id: 0}
  )
< {
    State: 'Mississippi',
    County: 'Newton County',
    Rate: 6.1
  }
  {
    State: 'Mississippi',
    County: 'Monroe County',
    Rate: 7.9
  }
  {
    State: 'Mississippi',
    County: 'Hinds County',
    Rate: 6.1
  }
  {
```
(too many results)
    d. Explanation:
        i. This query uses a range filter ($gte and $lte) to find unemployment rates between 5% and 8%, inclusive. The find method also projects only the relevant fields for concise results.

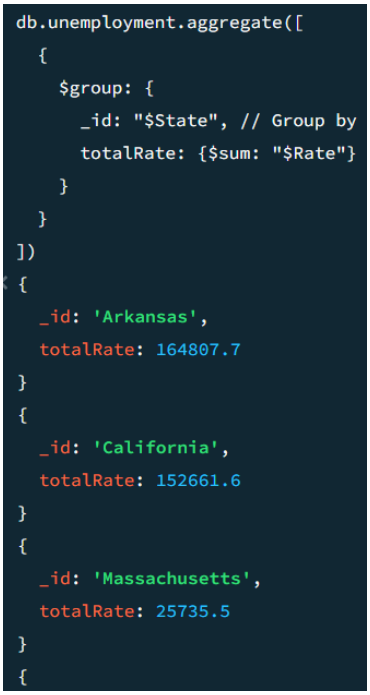
7. Find the state with the highest unemployment rate. Hint. Use { $limit: 1 }
    a. Sort Rate in descending order while limiting to the first result.
    b. Query:
        i. db.unemployment.aggregate([
        ii. { $sort: {Rate: -1} },
        iii. { $limit: 1 },
        iv. { $project: {State: 1, Rate: 1, _id: 0} }
        v.])
    c. Screenshot:
        i.

```
> db.unemployment.aggregate([
    {
      $sort: {Rate: -1} // Sort records in 
    },
    {
      $limit: 1 // Limit the results to the
    },
    {
      $project: {State: 1, Rate: 1, _id: 0}
    }
  ])
< {
    State: 'Colorado',
    Rate: 58.4
  }
```

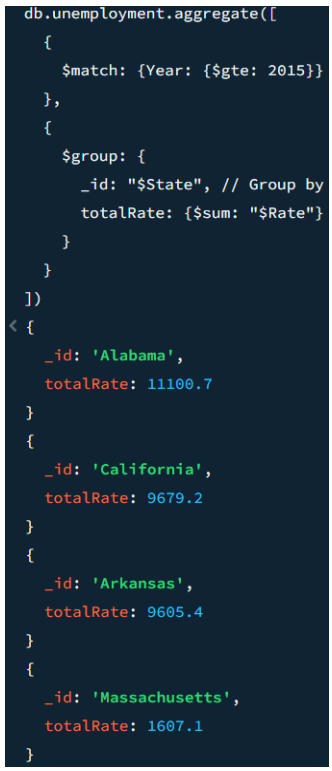
    d. Explanation:
        i. The dataset is sorted in descending order by Rate ($sort: {Rate: -1}), and the $limit: 1 stage ensures only the top record (highest rate) is returned.

8. Count how many counties have an unemployment rate above 5%.
    a. Find the total number of counties where Rate is greater than 5.
    b. Query:
        i. db.unemployment.find({Rate: {$gt: 5.0}}).count()
    c. Screenshot:
        i.
```
> db.unemployment.find({Rate: {$gt: 5.0}}).count()
< 510173
```
    d. Explanation:
        i. The query filters records where Rate is greater than 5% and uses .count() to return the total number.

9. Calculate the average unemployment rate per state by year.
    a. Find the total number of distinct years.
    b. Query:
        i. db.unemployment.aggregate([
        ii. { $group: {
        iii. _id: {State: "$State", Year: "$Year"},
        iv. averageRate: {$avg: "$Rate"}
        v. } },
        vi. { $sort: {"_id.Year": 1, "_id.State": 1} }
        vii.])
    c. Screenshot:
        i. 
```
> db.unemployment.aggregate([
    {
      $group: {
        _id: {State: "$State", Year: "$Year"}, // Group by State and Year
        averageRate: {$avg: "$Rate"} // Calculate the average of the Rate field
      }
    },
    {
      $sort: {"_id.Year": 1, "_id.State": 1} // Sort by Year and State for clarity
    }
  ])
< {
    _id: {
      State: 'Alabama',
      Year: 1990
    },
    averageRate: 8.226990049751244
  }
  {
    _id: {
      State: 'Arizona',
      Year: 1990
    },
    averageRate: 8.285555555555556
  }
```
(too many results)
    d. Explanation:
        i. This groups data by State and Year (_id: {State: "$State", Year: "$Year"}) and calculates the average unemployment rate for each.

10. For each state, calculate the total unemployment rate across all counties (sum of all county rates).
    a. Find the sum of Rate.
    b. Query:
        i. db.unemployment.aggregate([
        ii. { $group: {
        iii. _id: "$State",
        iv. totalRate: {$sum: "$Rate"} } }
        v.])
    c. Screenshot:
        i. 

(too many results)
    d. Explanation:
        i. This groups records by State and calculates the sum of unemployment rates ($sum: "$Rate") for all counties within each state.

11. The same as Query 10 but for states with data from 2015 onward.
    a. Find the sum of Rate for states with data from 2015 and later.
    b. Query:
        i. db.unemployment.aggregate([
        ii. { $match: {Year: {$gte: 2015}} },
        iii. { $group: {
        iv. _id: "$State",
        v. totalRate: {$sum: "$Rate"} } }
        vi.])
    c. Screenshot:
        i. 
```
db.unemployment.aggregate([
  {
    $match: {Year: {$gte: 2015}}
  },
  {
    $group: {
      _id: "$State", // Group by
      totalRate: {$sum: "$Rate"}
    }
  }
])
< {
  _id: 'Alabama',
  totalRate: 11100.7
}
{
  _id: 'California',
  totalRate: 9679.2
}
{
  _id: 'Arkansas',
  totalRate: 9605.4
}
{
  _id: 'Massachusetts',
  totalRate: 1607.1
}
```
(too many results)
    d. Explanation:
        i. Same as query 10 but the $match stage filters records for years 2015 or later.