**CSP-554**

**Evaluation Of COVID-19 Trends and Safety Measures - Project Report**

# Table of Contents

[**Table of Contents**](#_6100gw7noull) **1**

[**Team Contributions**](#_u1lkjq51h6bs) **2**

[**Abstract**](#_794m2xdu44f) **2**

[**Introduction**](#_er4hlkuqs1rn) **2**

[**Literature Review**](#_izjjah3r8ti6) **3**

[**Data Description**](#_x3z3wumlkbov) **5**

[**Spark & R Preprocessing**](#_misq6gshw191) **6**

[**MongoDB Database Queries**](#_tbicfeav5if6) **7**

[**Interactive Visualization Dashboard**](#_5wbv3vnyp5ku) **16**

[**Results and Future Work**](#_ylwb5fr8awdd) **23**

[**References**](#_w36rishxmscg) **24**

[**Appendix A**](#_p89j3g8d3yvz) **25**

[**Appendix B**](#_bnzfjvmb71i) **27**

[**Appendix C**](#_92by79iyhk8e) **31**

# 

# Abstract

Given the scale of COVID-19 it’s clear that multiple institutions such as Johns Hopkins University would keep track of all the data regarding the pandemic. There exist multiple data repositories filled with information on factors such as hospitalization rates, death rates, and the implementations of state and nation-wide policies implemented to fight against the spread of the virus. In this paper we focus on finding the most versatile sets of data, preprocessing them, and then using multiple big data technologies such as Spark Dataframes and MongoDB to form insights on humanity's one year with COVID-19. For the key takeaways, we find that around December 2020 an increase in recovery rate can be noticed, possibly thanks to the rollout of vaccines, as well as general exponential increase in testing.

# Introduction

The COVID-19 pandemic has taken the entire world by surprise, continuing to be an extremely pressing issue in many countries. However, while the impact of this pandemic can be classified as nothing short of world-shaking, governments around the world have embraced varying responses regarding public health policies. Over time, the severity of these policies evolved along with the “seriousness” of the COVID situation within the area. The United States has been a unique entity throughout the pandemic as it has relied on individual state policies, rather than federal, to properly handle the virus.

This paper will focus particularly on the U.S. and explore the various state policy implementations over time. The goal here is to see if we can derive any useful insights about the effectiveness of particular policies. More specifically, we will compare the severities of individual state policies over time to various COVID-19 statistics, such as deaths, hospitalization rate, and so on. Because the states were impacted by the pandemic in varying levels, and because many states had varying responses due to the general absence of the federal government response, this is a great case study to observe when trying to evaluate policy effectiveness.

With the development and mass distribution of the COVID-19 vaccine, the impact of the pandemic has steadily declined in the United States. However, we must caution that research in this area is still ongoing, so confounding factors do exist. Much of the decline in areas like mobility, economic activity, and social interaction can also be attributed to private decisions made by individuals, regardless of the adoption of particular public health policies [7]. Confounding factors such as these are difficult to pinpoint since the situation is still fresh and ongoing. Despite this, it is important to identify which policies for combating COVID-19 are potentially more effective than others, given the set circumstances of a particular area.

# Literature Review

*Prem, K., Liu, Y., Russell, T. W., Kucharski, A. J., Eggo, R. M., Davies, N., ... & Klepac, P. (2020). The effect of control strategies to reduce social mixing on outcomes of the COVID-19 epidemic in Wuhan, China: a modelling study. The Lancet Public Health, 5(5), e261-e270.*

This article goes into the effects of population mixing in regards to the spread of COVID-19 epidemic. [5] In this context, population mixing includes closures of schools, workplaces, and other public locations. To perform this examination, authors of the study used a SEIR model, transmission models, and ran simulations against the collected data. All those methods are described in depth with equations and visualizations attached, which will help us in creating our final insights and conclusions based on the data that we process ourselves. On top of that, since the findings in the article - similar to our own goals - touch upon the effectiveness of policies like school closures, we will be able to compare our results against these. This will further help measure the continuous effect of these policies since the article was published on March 25th 2020, while we will be analysing time series data going all the way through the 30th of November 2020.

*Hsiang, S., Allen, D., Annan-Phan, S., Bell, K., Bolliger, I., Chong, T., ... & Wu, T. (2020). The effect of large-scale anti-contagion policies on the COVID-19 pandemic. Nature, 584(7820), 262-267.*

Similar to the previous one, this article discusses policies designed to slow down the rate of infections, once again including closures and travel restrictions. [2] In the case of this study, data was collected from multiple countries including China, United States, Italy and France. Some of the key takeaways from this paper include the fact that different policies had different effects on unique populations but in general they all seem to slow down the spread of COVID-19. Since our own data is collected on a state level from the United States, we can compare and contrast our results against the ones in this research since they also have state level granular data. The visualizations within this article involve variables similar to the ones we are using (infection cases and deaths), giving us yet more examples of how to present and analyze our data.

*Castillo, R. C., Staguhn, E. D., & Weston-Farber, E. (2020). The effect of state-level stay-at-home orders on COVID-19 infection rates. American Journal of Infection Control, 48(8), 958-960.*

This entry in the American Journal of Infection Control goes in depth into the stay-at-home orders carried out by different states within the United States and their effectiveness against the spread of COVID-19. [1] Even though the way in which the study was conducted does include some limitations (for example, not being able to isolate the effectiveness of these orders against other policies being put into place at the same time), the general consensus is that these policies did indeed help with slowing down the infection rates. This study used sensitivity analysis and linear regression models to obtain the results, which is extremely helpful in our case since for the data analysis piece we are also using linear regression to measure different policies against each other. We also possess data on the stay-at-home orders, making this research a great resource for validation.

*Krome, C., & Sander, V. (2018). Time series analysis with apache spark and its applications to energy informatics. Energy Informatics, 1(1), 337-341.*

This article discusses the implementation and effectiveness of using Apache Spark and R for time series data and analysis, in particular a German day-ahead spot market. [3] It introduces and explains time series methods used within the energy industry for forecasting, and shows how Apache Spark can be used to perform these calculations and analysis in parallel. While we ourselves aren’t working with energy informatics, our COVID-19 data is also a time series, and therefore this research helps us understand how to handle it. Since we are using Apache Spark and R throughout our entire project (both for data preprocessing and for analysis), we can use this article as a starting point on how to use these technologies for our time series analysis.

*Zhang, X., Saleh, H., Younis, E. M., Sahal, R., & Ali, A. A. (2020). Predicting Coronavirus Pandemic in Real-Time Using Machine Learning and Big Data Streaming System. Complexity, 2020.*

This research article explains how the Twitter Streaming API, Apache Kafka and Apache Spark were used to predict COVID-19 in real time. [6] Using data from Twitter collected between January and June of 2020, the authors used machine learning algorithms such as decision trees, logistic regression, k-nearest neighbors and others to find the best model for prediction. This write-up explains how the data was processed through Kafka and Spark, and why in the end the random forest machine learning algorithm turned out to be the best choice. While the purpose of this research is different to ours (prediction vs. analysis), the detailed explanations on their use of Apache Spark and the general information on COVID-19 data is powerful and extremely useful to our case.

*Ponce, M., & Sandhel, A. (2020). covid19. analytics: An R Package to Obtain, Analyze and Visualize Data from the Corona Virus Disease Pandemic. arXiv preprint arXiv:2009.01091.*

This paper goes over an R package created to obtain and analyze data on the COVID-19 pandemic. [4] The data that can be accessed through it is worldwide and includes built-in tools for analysis and visualizations. Additionally, the data is split into multiple types for ease of access, including time series and aggregated data. The sources for the data overlap with the ones that we are using for our project, that being the John Hopkins University data. Therefore, when we begin to work on our visualizations and analysis in R, this package can really speed up the process. While most of our preprocessing work will be done in Apache Spark, R will definitely come in handy throughout the entire project, so having resources like this package will allow us to produce the best possible results.

# Data Description

For our project goals, we needed a dataset that had time series data for both policy information per every U.S. state and various COVID-19 statistics. The University of Oxford’s COVID-19 Government Response dataset for the United States was nearly perfect for our purposes. It contained time series data dating back to January 1st, 2020 up until the current day. This data included various categories of policies, including containment and closure policies such as school closing, restrictions on gatherings, and stay at home requirements, as well as health system policies such as contact tracing and facial coverings. What made this dataset even more ideal was that it categorized each change in policy on a severity scale. This made it much easier to be able to parse the various policy developments into easily interpretable groups of severity. The lowest severity of 0 implied no policy implemented; each policy category has its own scaling and classification. Specific details on the classification of each severity level for every policy can be found on the Oxford dataset Github page.

The Oxford dataset also had two COVID-19 statistics columns: one for the number of cumulative confirmed cases and one for the number of cumulative deaths from the pandemic. However, this dataset lacked some vital statistics such as the testing rate, hospitalization rate, total recovered cases, etc. For these additional statistics we utilized the Johns Hopkins University COVID-19 Daily US State Reports dataset. While this dataset contained several other statistics that we can use, the main drawback of this dataset is that it was limited to data after 04/12/2020.

In the end, we ended up joining the two datasets and analyzing policy and statistics data from 04/12/2020 up to the current day. The data combination process, features kept, and other data cleaning/preprocessing was completed in R and translated into Spark. These details will be covered in the following section.

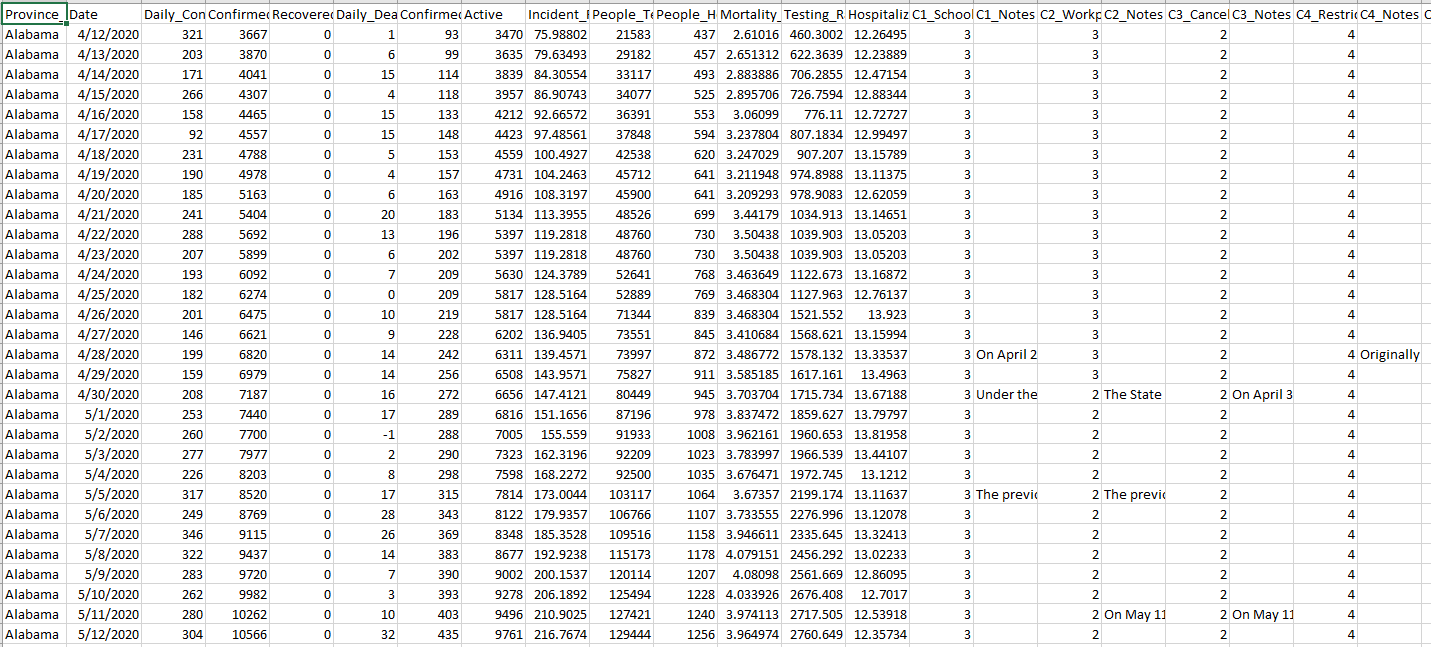
# Spark & R Preprocessing

As mentioned previously, two separate data repositories were used to create our final dataset that all the computations and visualizations were built off of. The first dataset will be referred to as John Hopkins University (JHU), and the second will be Oxford (OX). All of the preprocessing was done programmatically, except for the very first step. This step consisted of splitting the JHU dataset manually into two parts. The reason for this is that the JHU github repository changed names of two columns on 11/09/2020, which made it difficult to merge all of their .csv files into one dataframe programmatically. Therefore, the first step consisted of downloading all of the relevant JHU (<https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports_us>) and OX (<https://github.com/OxCGRT/USA-covid-policy/tree/master/data>) data from their respective github repositories, and then splitting the JHU .csv files into two separate folders - one with data before 11/09/2020, and one after. The entire downloaded dataset ranged from 01/01/2020 till 04/30/2021.

From this point, all the changes were done programmatically, and the respective R code can be found in Appendix A. The translated Spark Dataframes code which enables the full potential of big data technologies and cluster computing can be found in Appendix B. For the future joining of all three folders, a column within the OX dataset had to be renamed. Column titled “RegionName” was renamed to “Province\_State” to match the respective column in JHU. After that, the two columns changed on 11/09/2020 in the second JHU folder were renamed. The changed names “Total\_Test\_Results” and “Case\_Fatality\_Ratio” were changed back to “People\_Tested” and “Mortality\_Rate” respectively, which allowed all the files from both folders to be merged into one large dataframe. After that in JHU, the column “Last\_Update” was renamed to “Date” for a more clear description and to match the naming convention in OX.

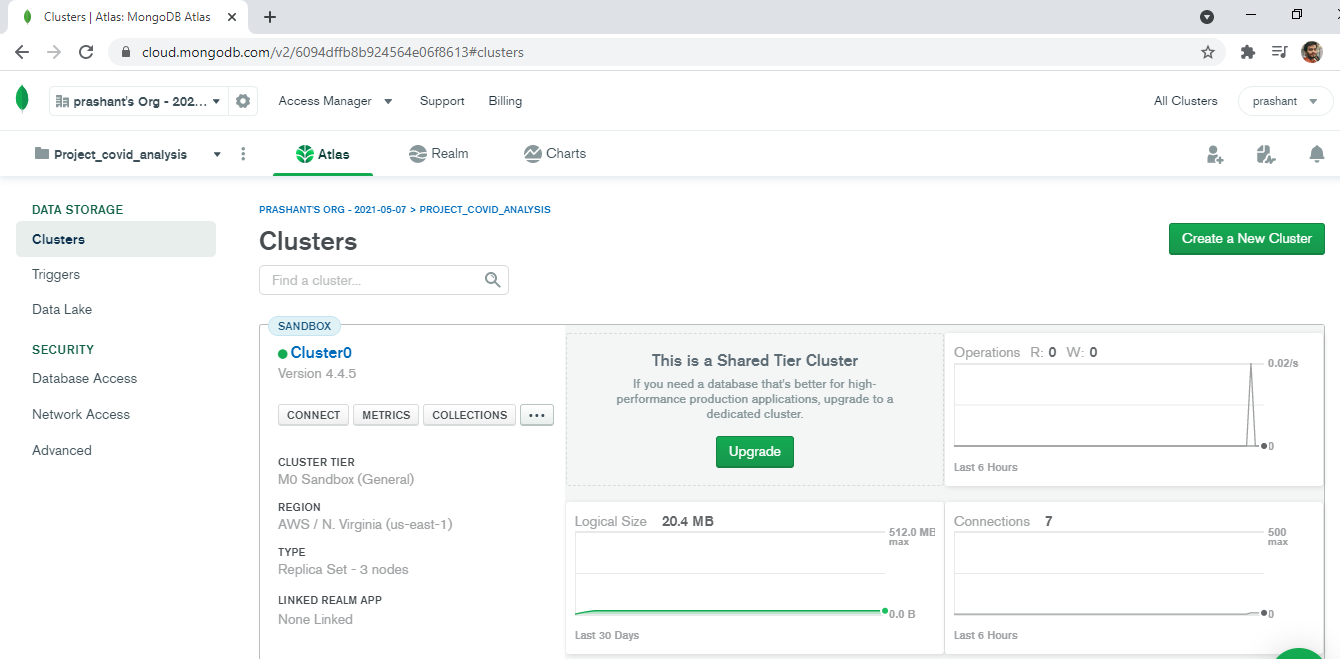
Next, fixes to data types in columns were made so that there's no issues when merging OX and JHU. The “Date” column in both was changed into R and Pyspark data types respectively. At that point, OX and JHU were merged into a dataframe codenamed “combined”. Combined was then ordered by “Province\_State'' and “Date” so that data can be easily searched through by states with information sorted chronologically. After that, unnecessary/garbage columns were removed, and the entire dataset was re-ordered. At that point, it was clear that there wasn’t much data available before the date of 04/12/2020, which makes sense since at that point the world was barely starting to understand the scale and importance of COVID-19. We decided to remove any entries in the data before that date, to allow for more accurate visualizations.

After some additional cleaning (like removing any row entries that weren’t U.S. states), there was one more final issue to tackle - null values. There were plenty of them considering that we merged two completely different github repositories, but in the end we managed to change all of them into logical values using filling and replacement with 0. In most cases, nulls were replaced with the last existing value before them in the column, and sometimes with the next existing value after them if relevant. Any other leftover null values were changed to 0’s since these small changes would not affect the results in a negative way. Screenshot of the final dataset can be seen below.

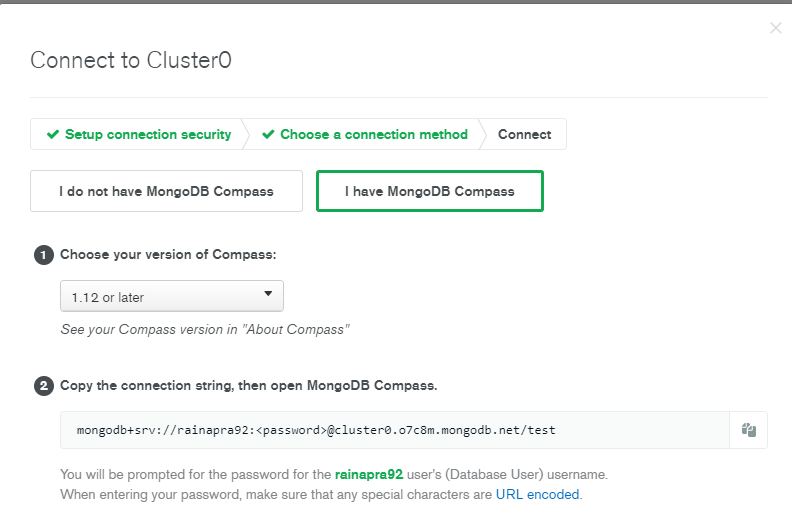


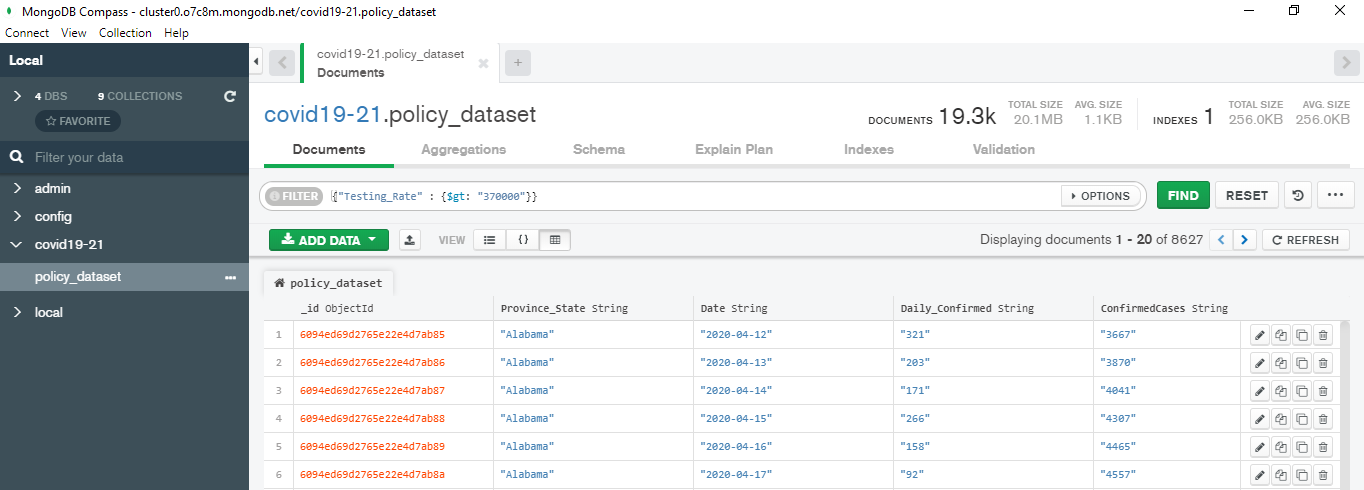
# MongoDB Database Queries

We have used MongoDB, NoSQL database to imply some queries in our COVID-19 policy dataset. The process can be seen below

Creating cluster in MongoDB

Connecting with that cluster using MongoDB compass to keep the dataset on cloud environment

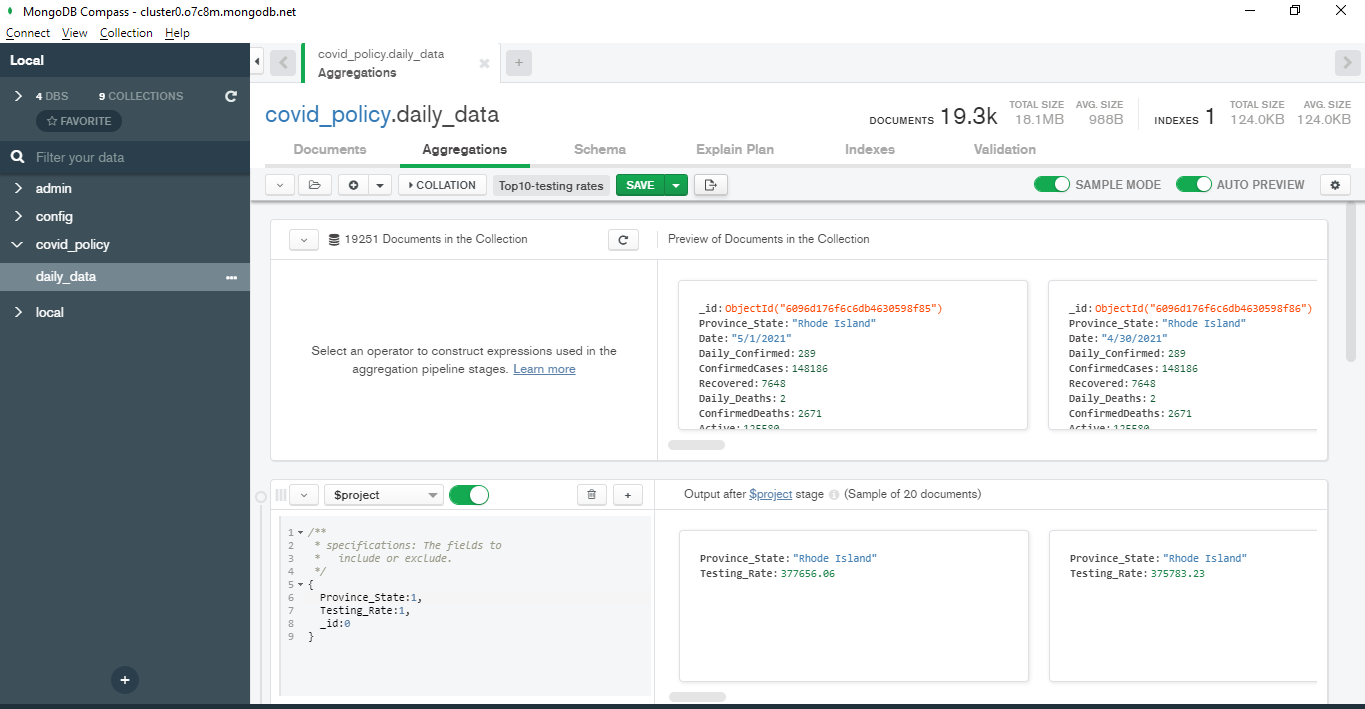


Ingesting the dataset into MongoDB compass using cloud connection

Following are some insights that we want to generate using MongoDB queries:

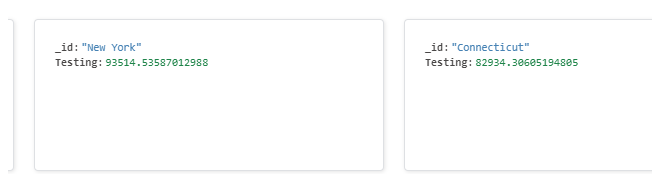
* Top 10 states with highest Testing rates.
* Top 10 states with lowest mortality rates.
* Top 10 states with highest number of people being recovered / Recovery rate.
* Top 10 highly affected states in terms of death counts.

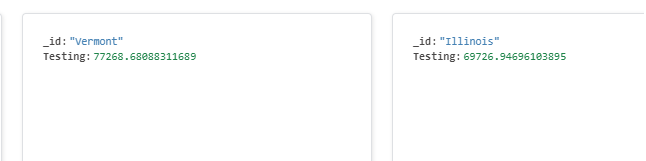
**Top 10 states with highest average testing rates**

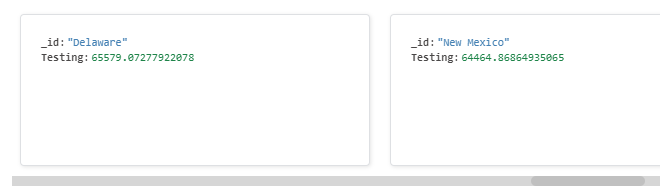
We have used the aggregation pipeline in the MongoDB compass to generate the results of the above queries. Python code written to perform those can be found in Appendix C. The process and results are below:









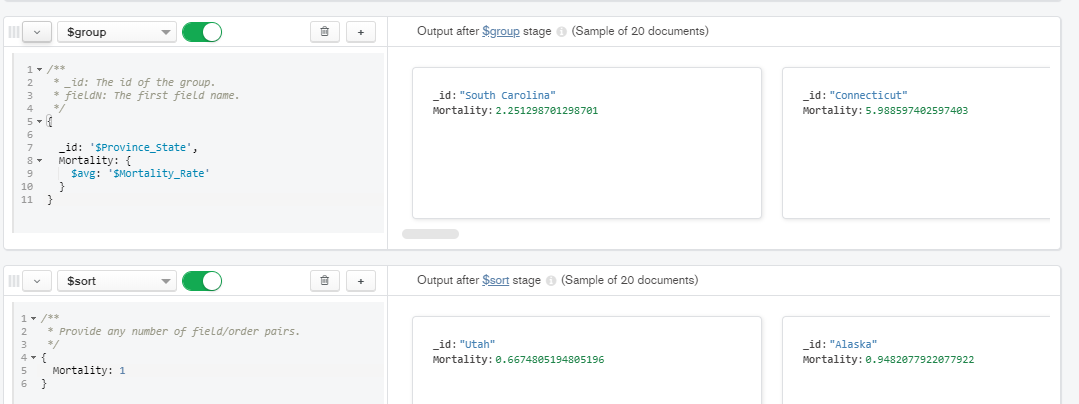


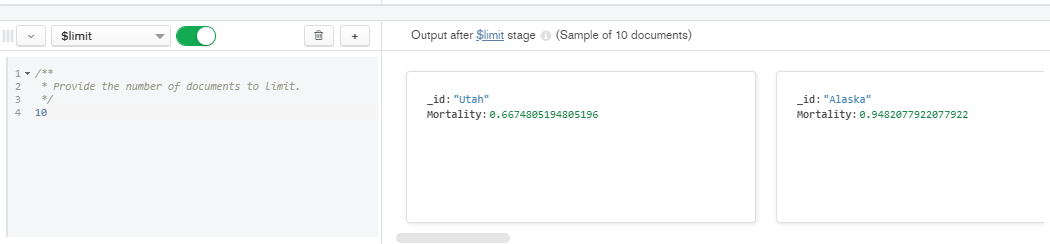
Below is the output of above pipeline in tabular format that displays the top-10 States and their average testing rates from highest to lowest:

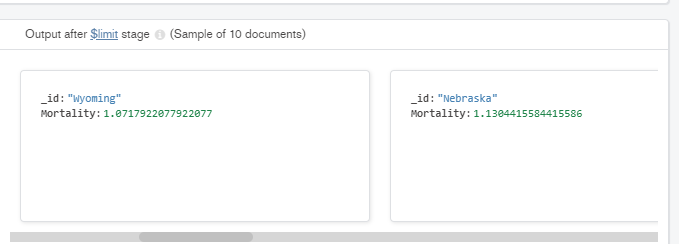
|  |  |
| --- | --- |
| **Province\_State** | **Average of Testing\_Rate** |
| Rhode Island | 130234.8278 |
| Alaska | 110659.7114 |
| Massachusetts | 106410.6882 |
| North Dakota | 96787.70119 |
| New York | 93514.5361 |
| Connecticut | 82934.3061 |
| Vermont | 77268.68084 |
| Illinois | 69726.94694 |
| Delaware | 65579.07281 |
| New Mexico | 64464.86911 |

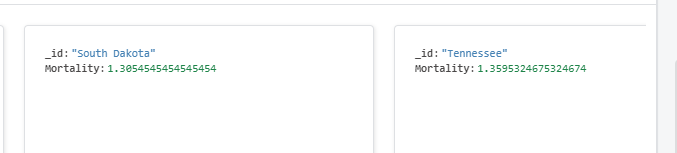
**Top 10 states with the average lowest mortality rates**

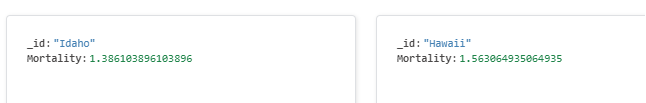
Similarly, we created the Aggregation pipeline to find the results of above query:

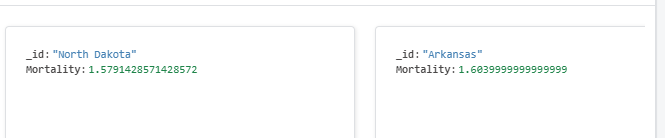










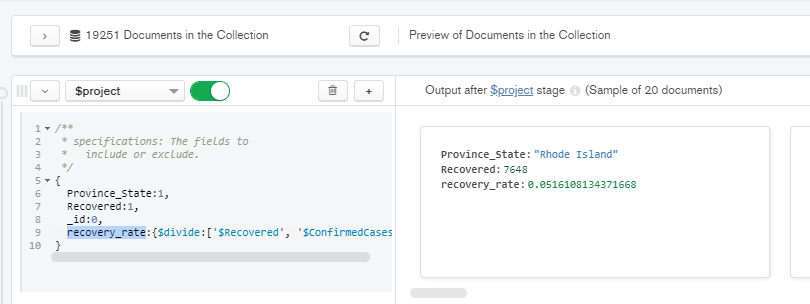


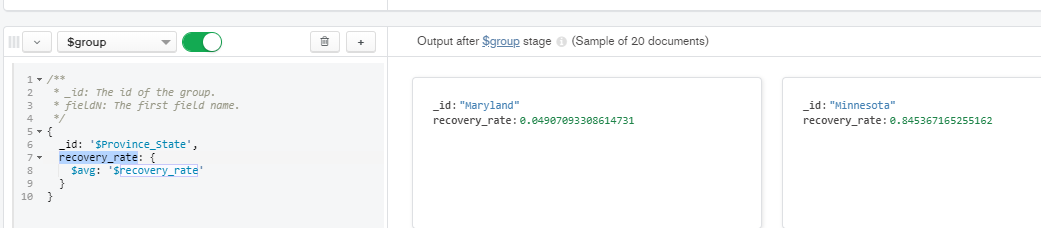
Below is the output of above pipeline in tabular format that displays the top-10 States and their average mortality rates from lowest to highest:

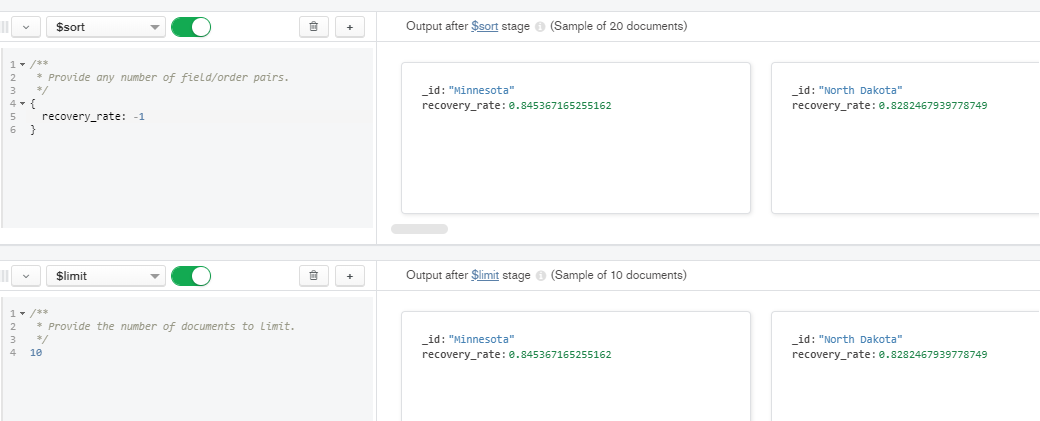
|  |  |
| --- | --- |
| **Province\_State** | **Average of Mortality\_Rate** |
| Utah | 0.66774985 |
| Alaska | 0.948172801 |
| Wyoming | 1.071867941 |
| Nebraska | 1.130364155 |
| South Dakota | 1.305258912 |
| Tennessee | 1.360008231 |
| Idaho | 1.386084353 |
| Hawaii | 1.563002638 |
| North Dakota | 1.578948298 |
| Arkansas | 1.603758528 |

**Top 10 states with highest number of recovery rate**

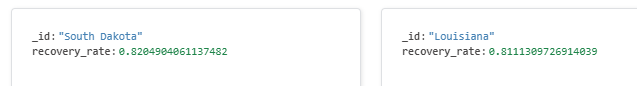
Similarly, we created the Aggregation pipeline to find the results of above query:



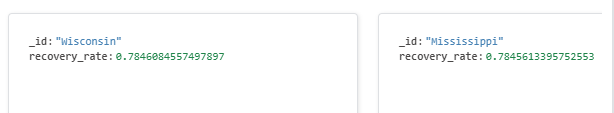










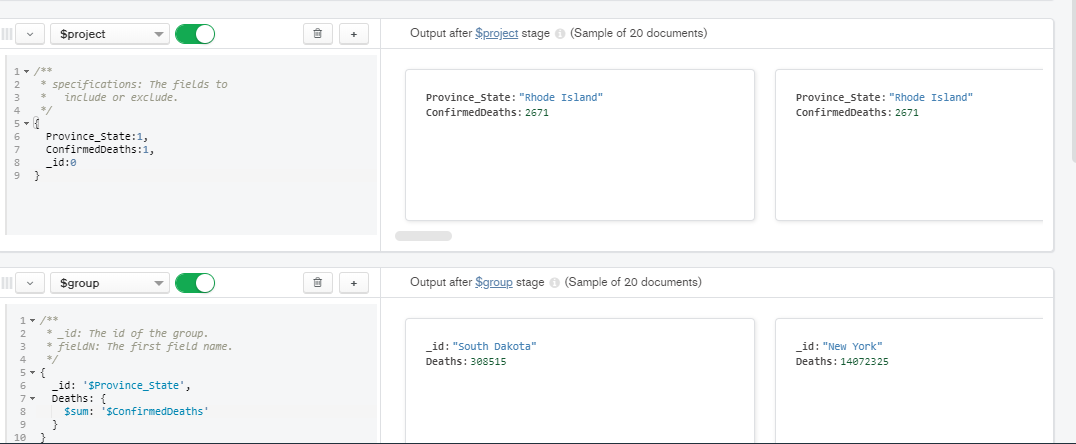


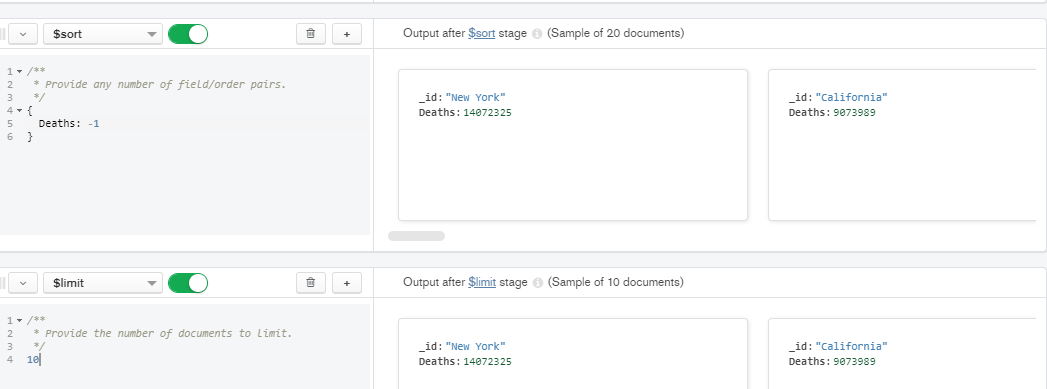
Below is the output of above pipeline in tabular format that displays the top-10 States and their average recovery rates from highest to lowest:

|  |  |
| --- | --- |
| **Province\_State** | **Average of Recovery\_Rate** |
| Minnesota | 0.8453671 |
| North Dakota | 0.8282467 |
| Oklahoma | 0.8257084 |
| Arkansas | 0.8214115 |
| South Dakota | 0.820490 |
| Louisiana | 0.8111309 |
| Wyoming | 0.8060983 |
| Tennessee | 0.7853631 |
| Wisconsin | 0.7846084 |
| Mississippi | 0.7845613 |

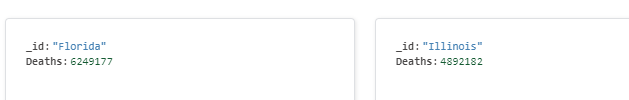
**Top 10 highly affected states in terms of death counts**

Similarly, we created the Aggregation pipeline to find the results of above query:













Below is the output of above pipeline in tabular format that displays the top-10 States and their absolute confirmed deaths from highest to lowest:

|  |  |
| --- | --- |
| **Province\_State** | **Sum of ConfirmedDeaths** |
| New York | 14072325 |
| California | 9073989 |
| Texas | 8119331 |
| New Jersey | 6704023 |
| Florida | 6249177 |
| Illinois | 4892182 |
| Pennsylvania | 4769662 |
| Massachusetts | 4127886 |
| Michigan | 3815272 |
| Georgia | 3399555 |

Major insights out of these MongoDB queries:

* Rhode Island has the highest average testing rate which means a large number of populations in that state are getting tested.
* Utah has the lowest average mortality rate which means a smaller number of deaths amongst the number of confirmed cases.
* Minnesota has the highest average recovery rate which means a large number of people getting recovered amongst the number of confirmed cases.
* New York has the highest number of death counts in total which makes it the highly affected state across the US.

# Interactive Visualization Dashboard

We have used Microsoft PowerBI for our visualizations.

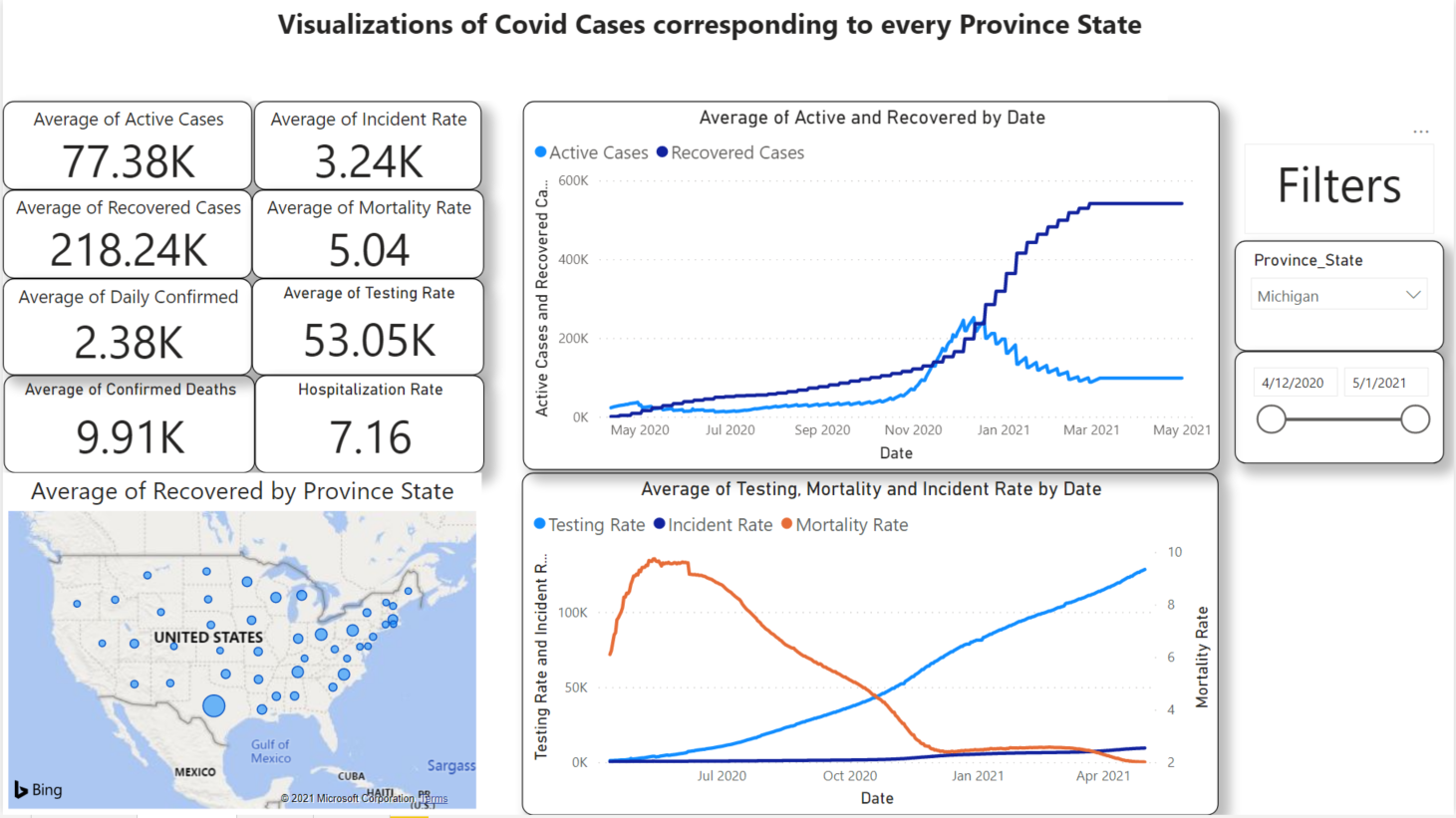
**Visualization 1 - Detailed analysis of Covid Cases corresponding to every Province State:**

We have created a fully interactive dashboard to infer the detailed covid analysis corresponding to every province state and in between the desired date range. In the below visualization, we have selected **Michigan** as our province state.

Line graph displaying the Average number of Active, Recovered, Daily confirmed cases, Mortality rate, Incident rate etc. corresponding to every province state.

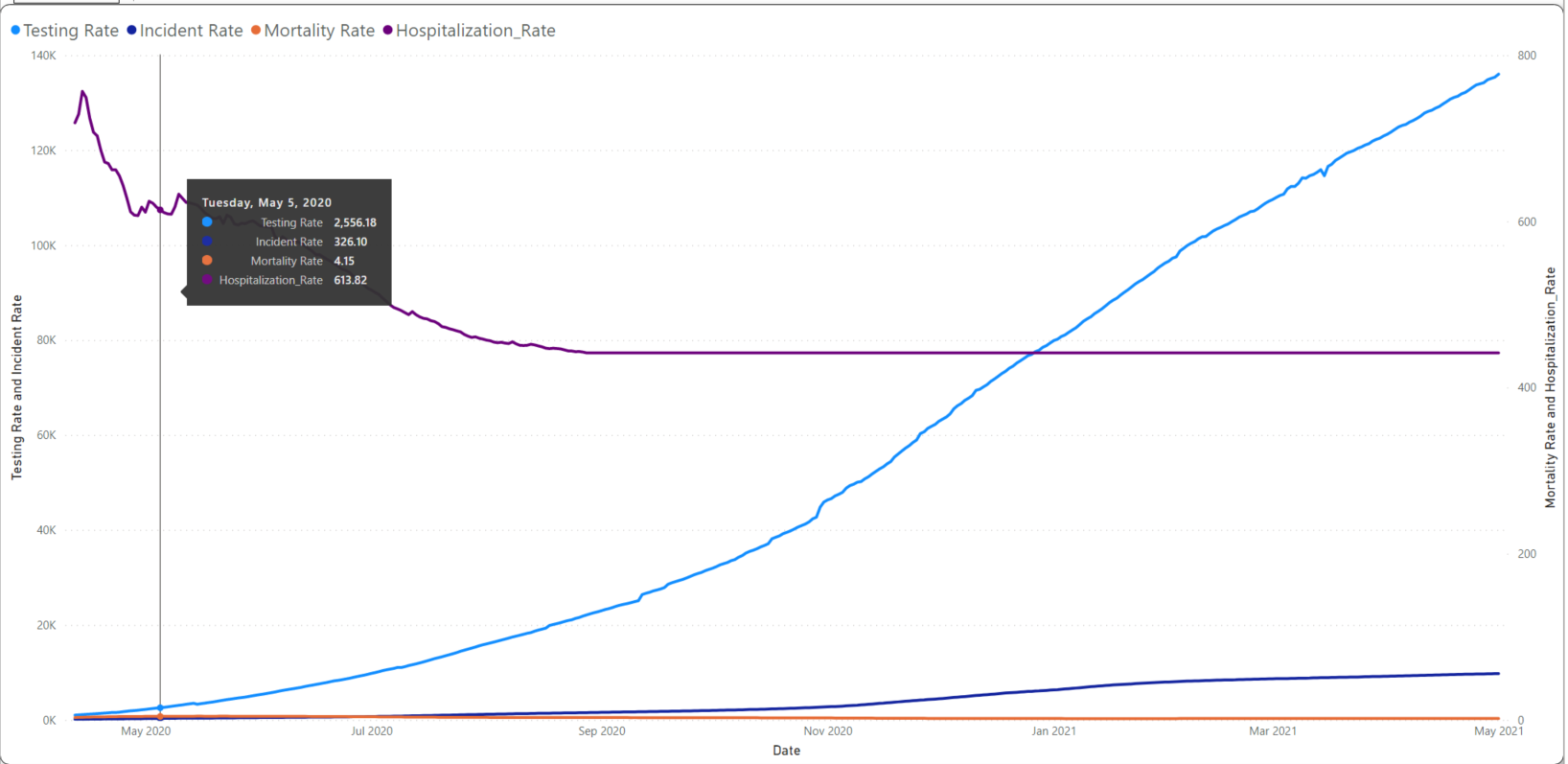
**Useful Insights:**

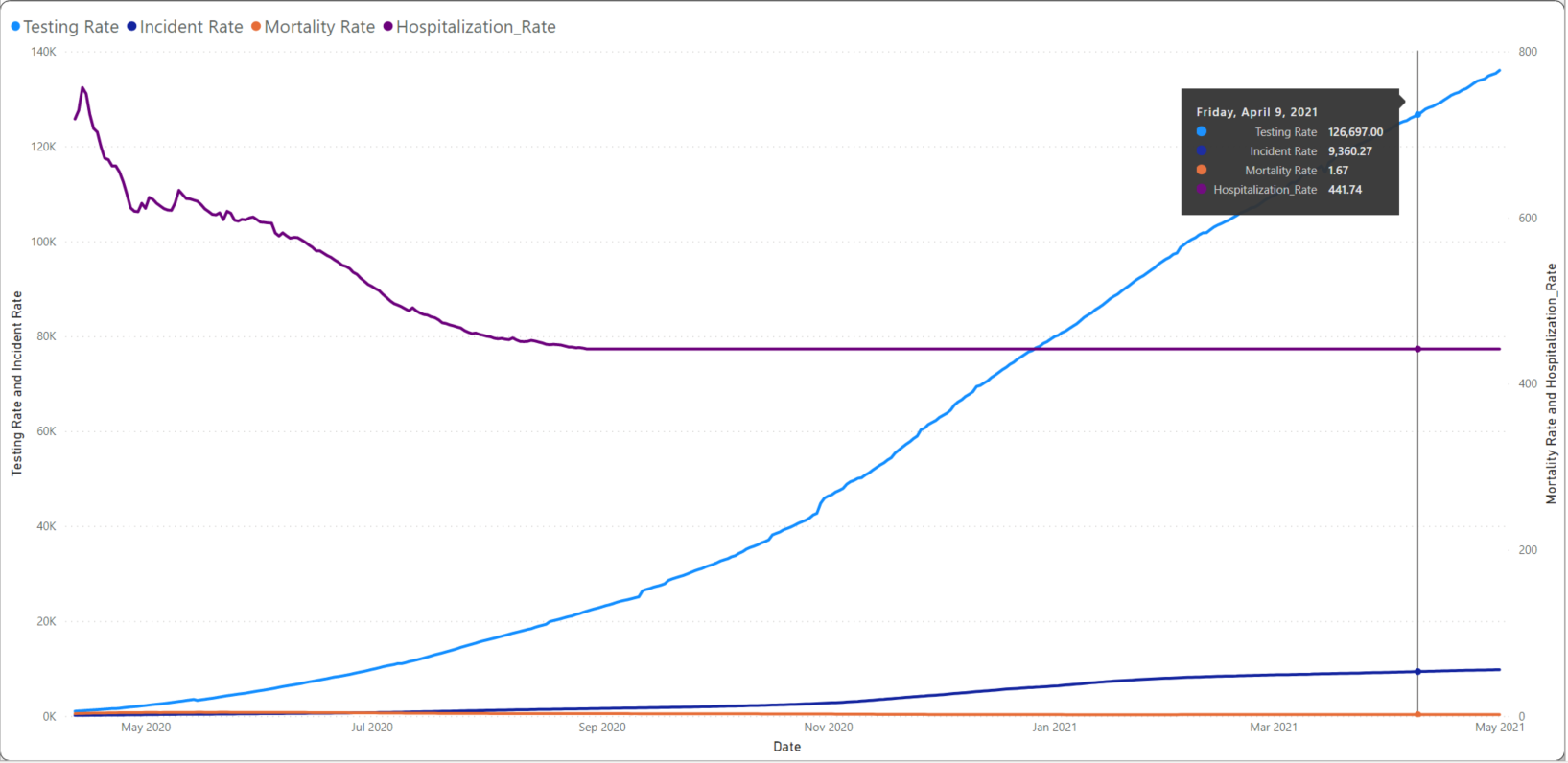
* Comparison of average active and recovered cases which shows that Active cases are more than the Recovered cases initially but eventually after Dec 2020, the situation started getting better with vaccination where Recovered cases are more than the Active cases.
* Comparison of Testing, Incident and Mortality rates corresponding to Michigan state. We can see that the Mortality rate was very high initially and improved dramatically with time. Testing Rate shows an exponential increase in the testing cases which is very great on the testing part whereas we can see that the Incident Rate is always increasing which is alarming.
* We can demarcate amongst states as per the average of recovered cases. We can see that Texas has shown the highest number of recovered cases.



**Visualization 2 - Comparison of average Testing, Mortality and Incident Rate by Date across US:**

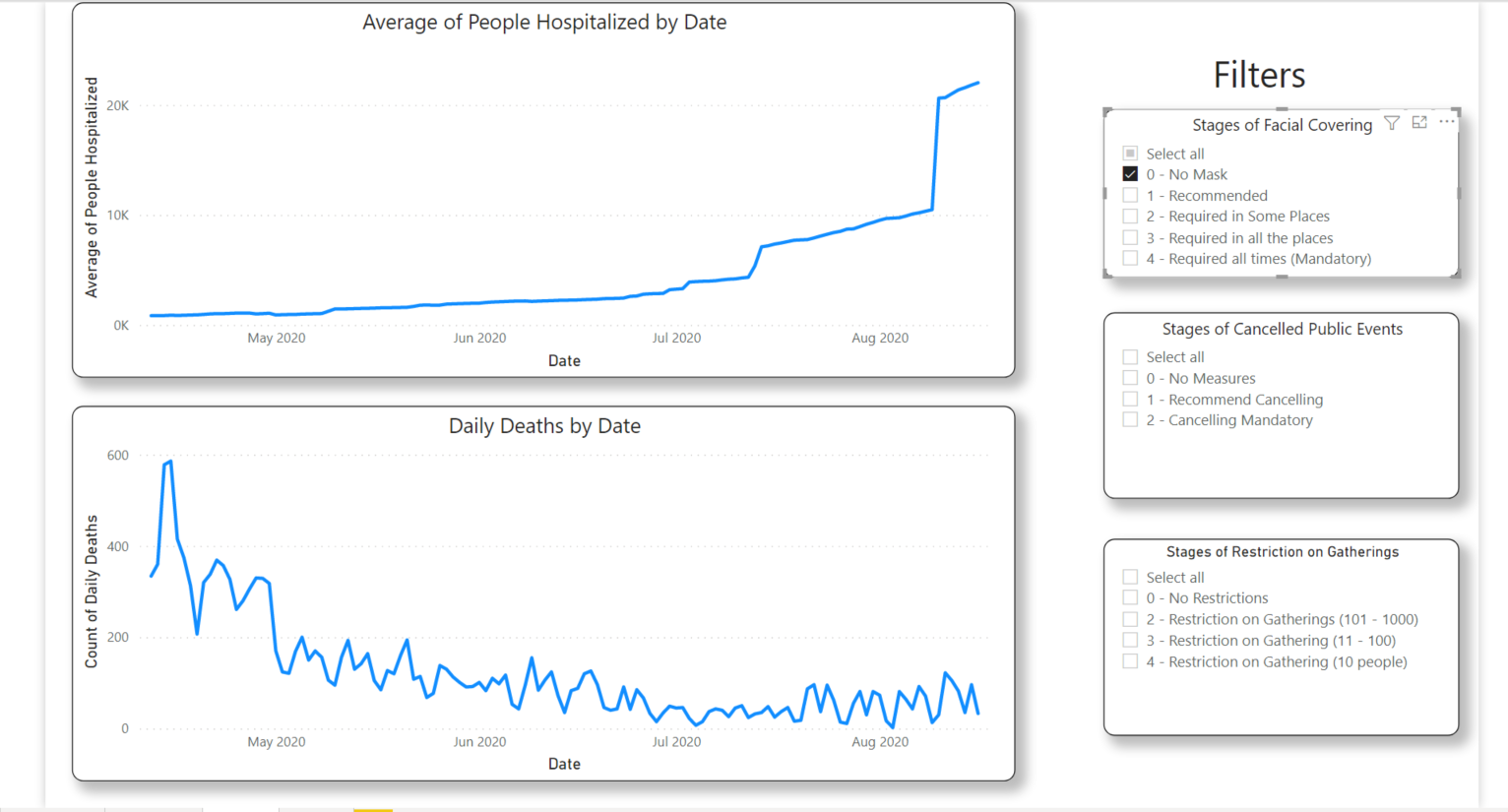
In the below visualization, we can see the trends in Testing, Incident, Mortality and Hospitalization Rates across the United States.

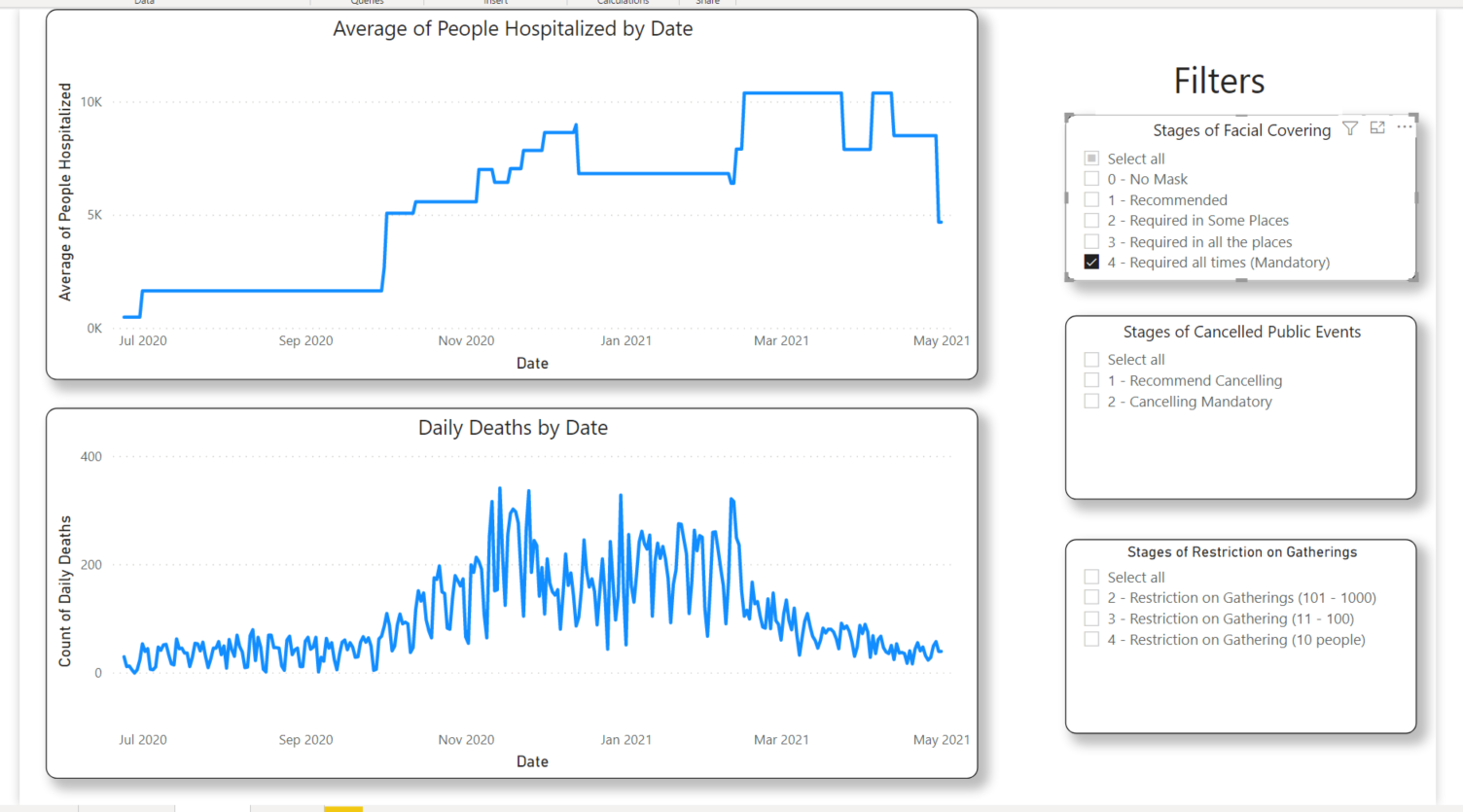




**Visualization 3 - Line graph displaying the number of people hospitalized and daily deaths before facial covering restriction:**

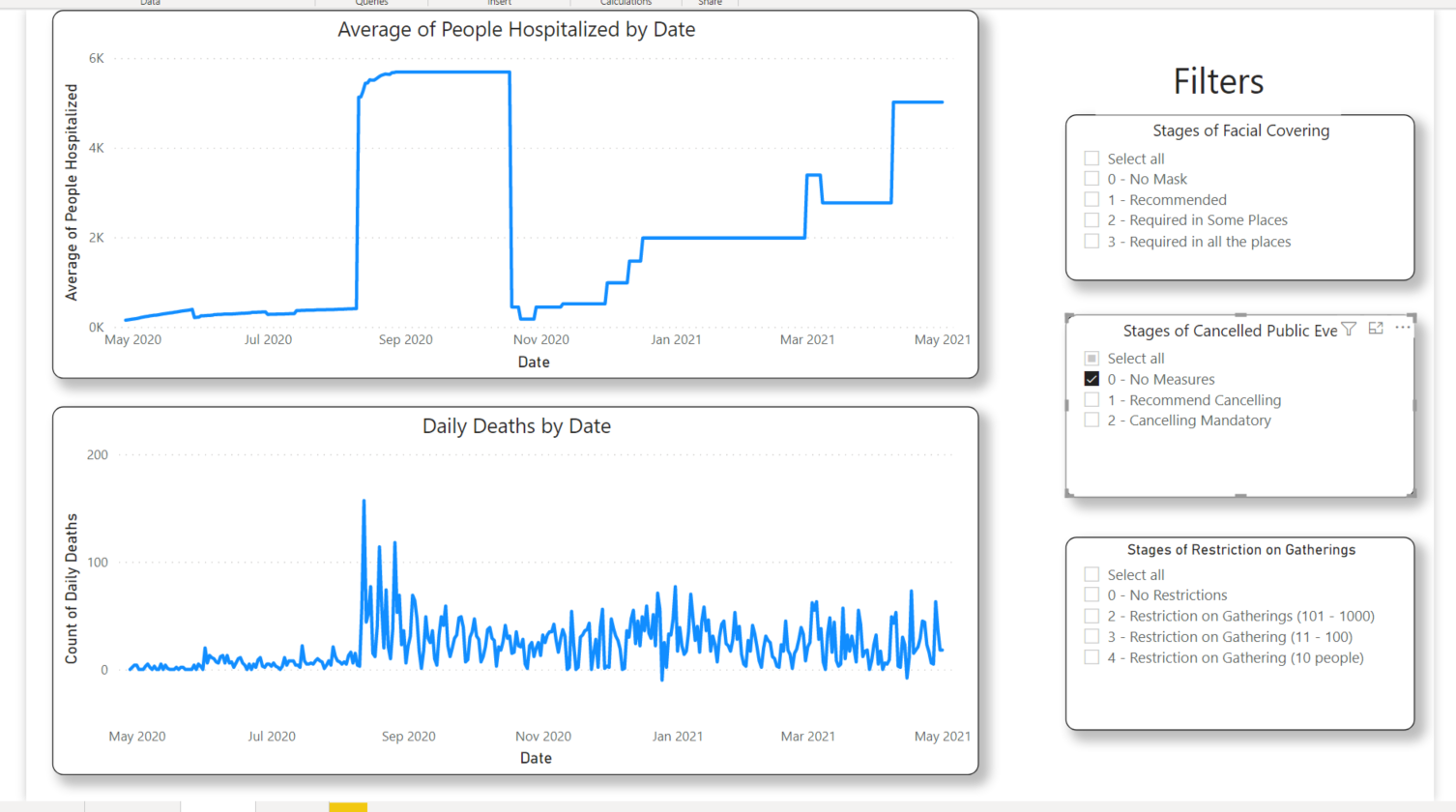
The below line graphs shows the changes in the number of patients hospitalized vs daily deaths by date when the government applied restrictions like Facial Covering, Cancelled Public Events and Restriction on Gatherings. We only compared restriction policies which were affecting the number of people getting hospitalized on the global level instead of the ones which were targeting any particular audience.  
  
People getting hospitalized before using masks were tremendously large in number and after implementing the mandatory usage of masks, we can see the decline in trend for the number of people hospitalized.

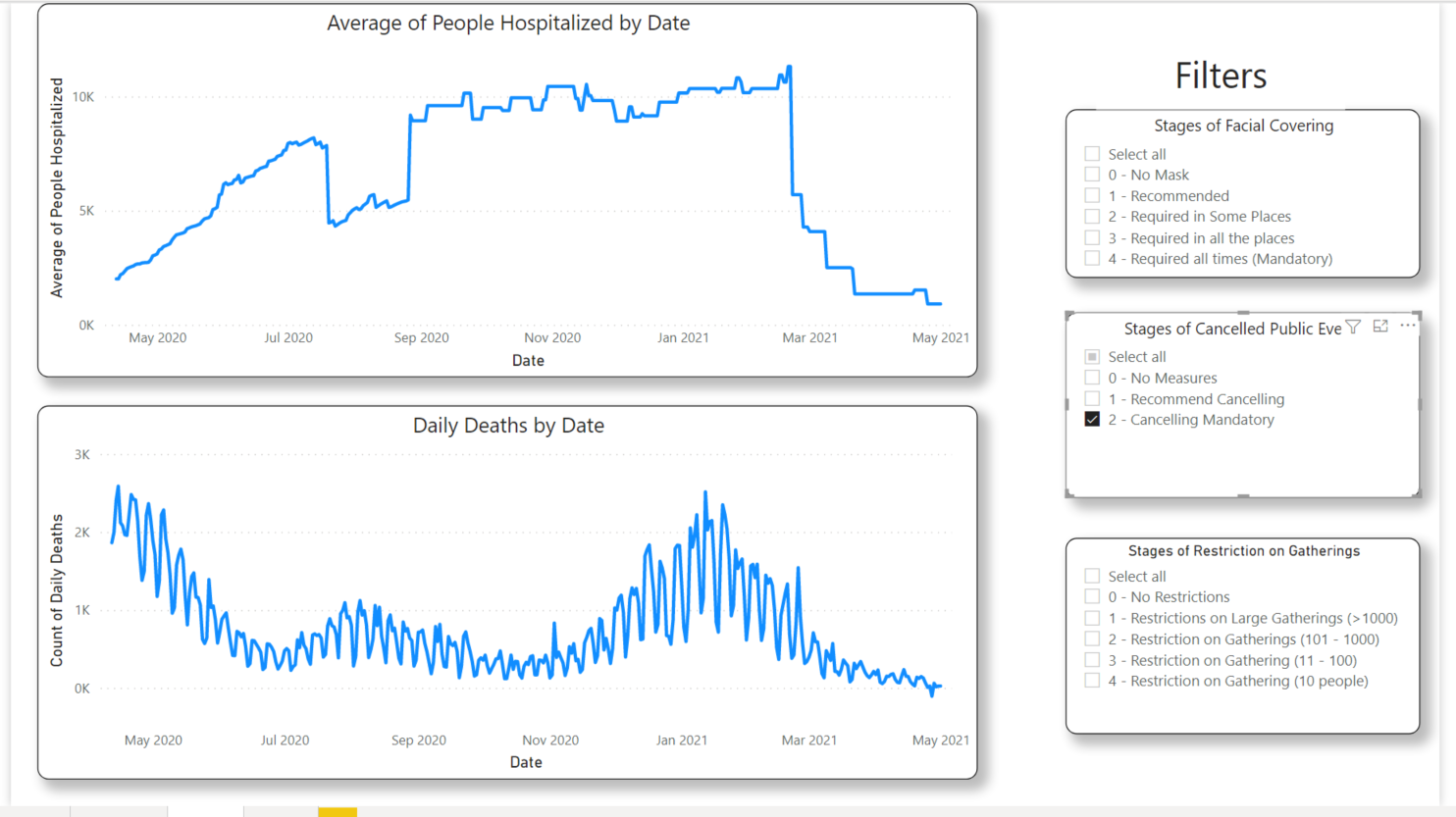
(Before Facial Covering Restriction)

(After Facial Covering Restriction)

**Visualization 4 -Line graph displaying the number of people hospitalized VS daily deaths before public events restriction:**

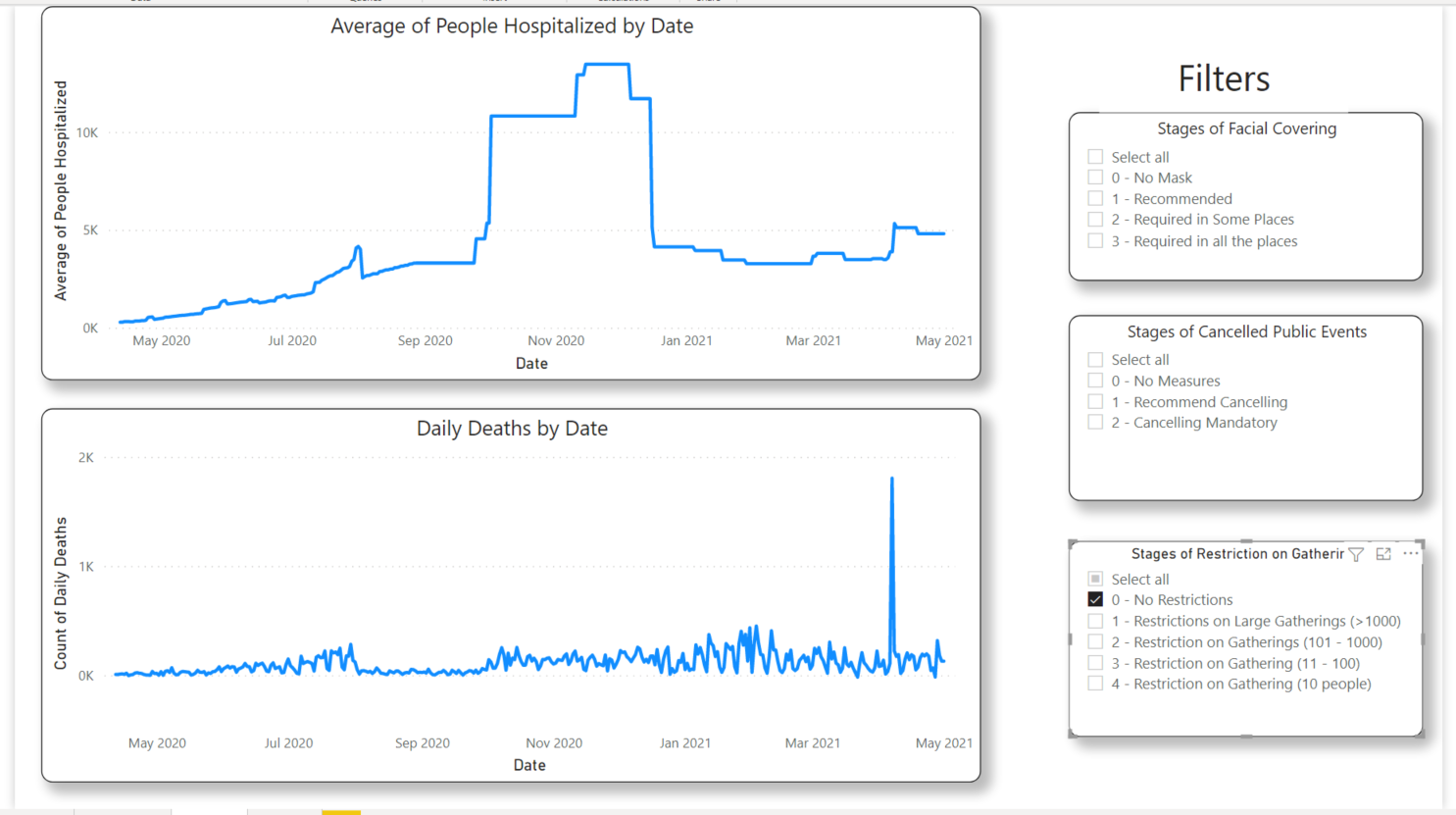
People getting hospitalized before public events restrictions were very large in number and increasing drastically but after implementing the mandatory restrictions on public events, we can see the decline in trend for the number of people hospitalized.

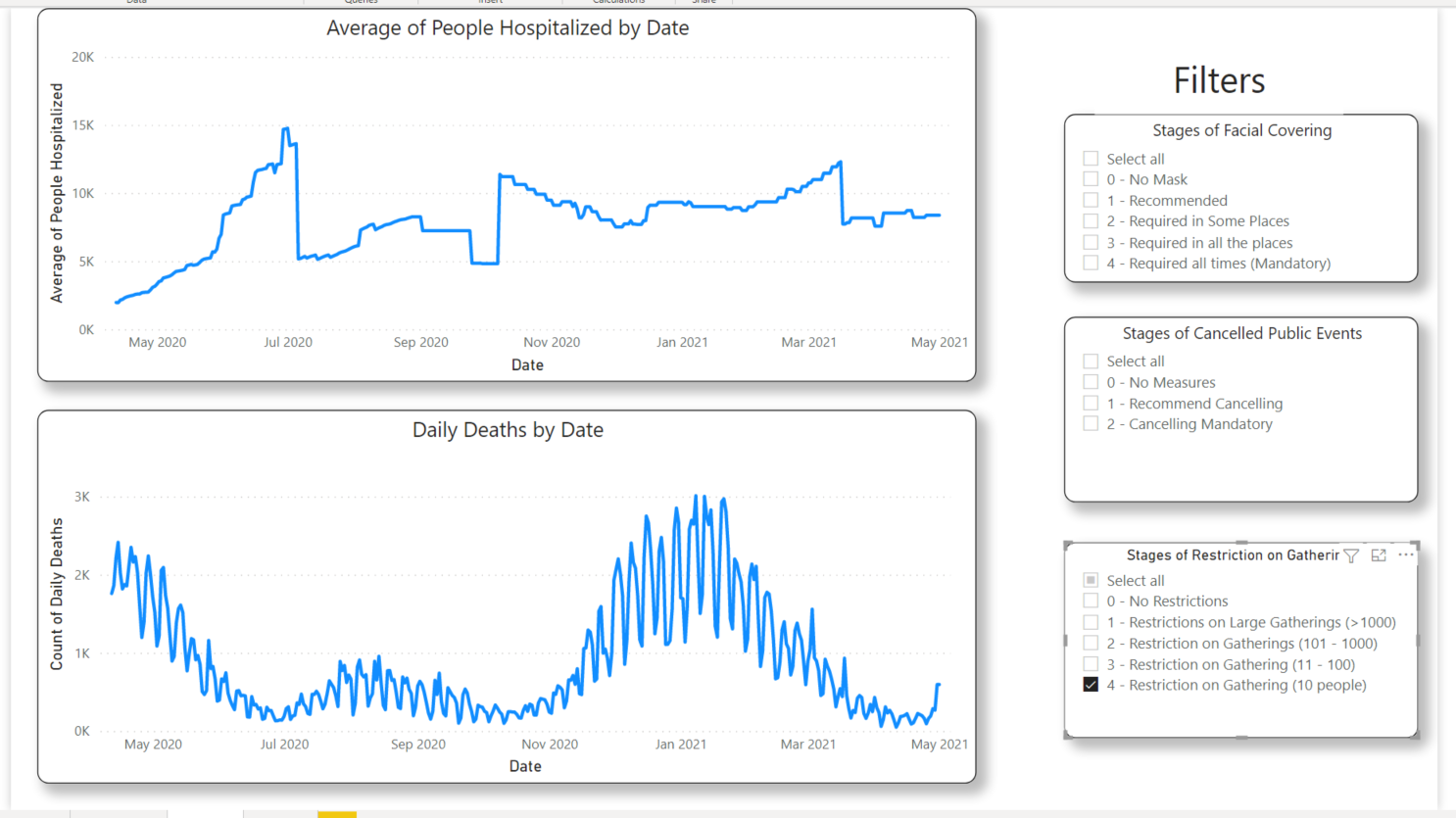
(Before Public Events Restriction)

(After Public Events Restriction)

**Visualization 5 - Line graph displaying the number of people hospitalized VS daily deaths before public gathering restriction:**

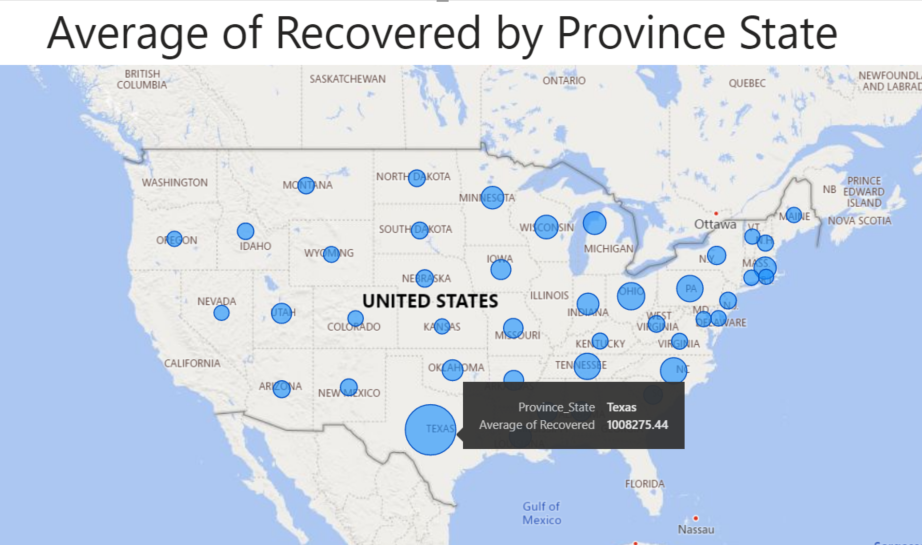
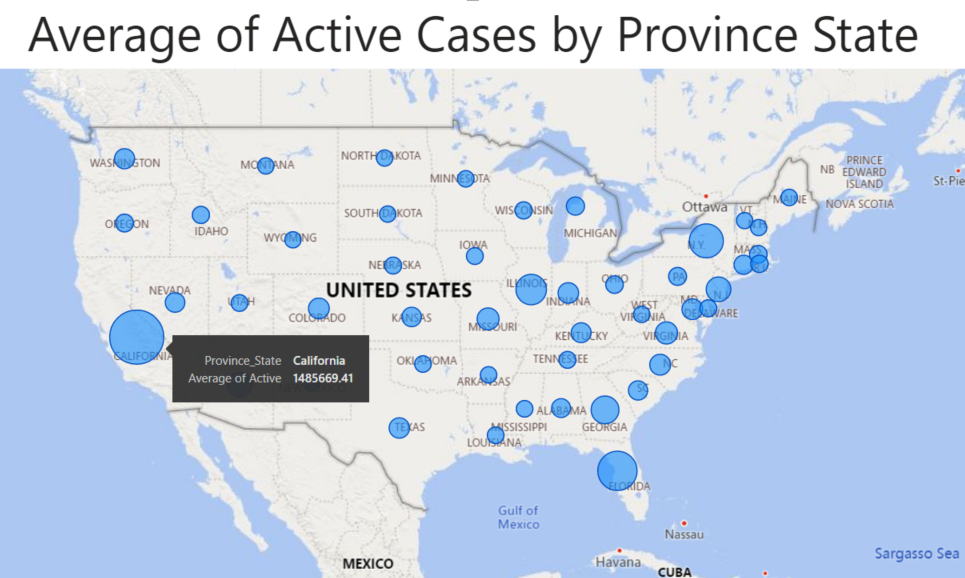
People getting hospitalized before public gathering restrictions were increasing drastically but after implementing the mandatory restrictions on public gathering, we can see the slight decline in trend for the number of people hospitalized.

(Before Public Gathering Restriction)

(After Public Gathering Restriction)  


**Visualization 6 - Following visualizations shows the difference in average number of active and recovered cases by Province States.**

California shows the highest number of Active cases whereas Texas shows the highest number of recovered cases.



# Results and Future Work

We used multiple big data technologies such as Spark Dataframes and MongoDB to preprocess and make insights using data pulled from github repositories of institutions like John Hopkins University and University of Oxford. The data showed us that every state widely differs in their approach towards the COVID-19 pandemic. For example, New York had the highest number of death counts in total, making it the most affected state across the US. And yet, they were not able to perform as much testing as states like Rhode Island or recoveries like Texas. As for more nation-wide statistics, by comparing average active and recovered cases we showed that around December 2020 the overall situation started getting better. Recovered cases started outpacing active cases, possibly thanks to the introduction of vaccinations.

This work can serve as a basis for future implementations of models. Although we had plans initially to implement some modeling using these datasets, the time constraints did not leave much room for model construction. Some potential models that can be constructed and analyzed include, but are not limited to the following:

* Regression model that analyzes the effectiveness of particular COVID-19 policies in relation to particular statistics. For example, the effectiveness of school closing policies with regard to the hospitalization rate. This effectiveness can be classified by the rise and fall of the hospitalization during each policy severity phase across every state.
* Predictive model that predicts the peak of the spikes of hospitalization rate. If an accurate forecast of the hospitalization can be made, this can help allocate the appropriate amount of resources such as ICU beds ahead of time for each state.
* Cluster model that predicts the likelihood of a spike in cases due to the severity of the situation in nearby states. This could help in developing policies that restrict travel from nearby states and further support the classification of states as “high risk”.

# 

# References

[1] Castillo, R. C., Staguhn, E. D., & Weston-Farber, E. (2020). The effect of state-level stay-at-home orders on COVID-19 infection rates. American Journal of Infection Control, 48(8), 958-960.

[2] Hsiang, S., Allen, D., Annan-Phan, S., Bell, K., Bolliger, I., Chong, T., ... & Wu, T. (2020). The effect of large-scale anti-contagion policies on the COVID-19 pandemic. Nature, 584(7820), 262-267.

[3] Krome, C., & Sander, V. (2018). Time series analysis with apache spark and its applications to energy informatics. Energy Informatics, 1(1), 337-341.

[4] Ponce, M., & Sandhel, A. (2020). covid19. analytics: An R Package to Obtain, Analyze and Visualize Data from the Corona Virus Disease Pandemic. arXiv preprint arXiv:2009.01091.

[5] Prem, K., Liu, Y., Russell, T. W., Kucharski, A. J., Eggo, R. M., Davies, N., ... & Klepac, P. (2020). The effect of control strategies to reduce social mixing on outcomes of the COVID-19 epidemic in Wuhan, China: a modelling study. The Lancet Public Health, 5(5), e261-e270.

[6] Zhang, X., Saleh, H., Younis, E. M., Sahal, R., & Ali, A. A. (2020). Predicting Coronavirus Pandemic in Real-Time Using Machine Learning and Big Data Streaming System. Complexity, 2020.

[7] Ziedan, E., Simon, K. I., & Wing, C. (2020). Effects of state COVID-19 closure policy on non-COVID-19 health care utilization (No. w27621). National Bureau of Economic Research.

# 

# Appendix A

**Preprocessing R Code**

|  |
| --- |
| if (!require("ggplot2")) install.packages("ggplot2")  library(ggplot2)  if (!require("zoo")) install.packages("zoo")  library(zoo)  if (!require("plyr")) install.packages("plyr")  library(plyr)  if (!require("dplyr")) install.packages("dplyr")  library(dplyr)  if (!require("tidyverse")) install.packages("tidyverse")  library(tidyverse)  #-----------------------------------------------------------  #Data filepaths  #Oxford  path\_ox = "D:/Kacper/School and Internship Work/Year 4 Sem 2/CSP 554 - Big Data Technologies/PROJECT/DATA/useful\_ox/OxCGRT\_US\_latest.csv"  #Johns Hopkins split into two files due to column name changes on 11/09/2020  path\_jhu\_1 = "D:/Kacper/School and Internship Work/Year 4 Sem 2/CSP 554 - Big Data Technologies/PROJECT/DATA/useful\_jhu/jhu1/"  path\_jhu\_2 = "D:/Kacper/School and Internship Work/Year 4 Sem 2/CSP 554 - Big Data Technologies/PROJECT/DATA/useful\_jhu/jhu2/"  path\_final\_write = "D:/Kacper/School and Internship Work/Year 4 Sem 2/CSP 554 - Big Data Technologies/PROJECT/DATA/final\_dataset.csv"  #-----------------------------------------------------------  #Oxford cleanup  policies <- read.csv(file= path\_ox, header=TRUE)  #policies <- subset(policies, select = -c(CountryName, CountryCode, RegionCode, Jurisdiction, 55:66))  names(policies)[names(policies) == "RegionName"] <- "Province\_State"  #-----------------------------------------------------------  #JHU cleanup  setwd(path\_jhu\_1)  file\_list <- list.files(path\_jhu\_1)  ldf <- lapply(file\_list , read.csv)  df.final <- do.call("rbind", ldf)  setwd(path\_jhu\_2)  file\_list2 <- list.files(path\_jhu\_2)  ldf2 <- lapply(file\_list2 , read.csv)  df.final2 <- do.call("rbind", ldf2)  names(df.final2)[names(df.final2) == 'Total\_Test\_Results'] <- 'People\_Tested'  names(df.final2)[names(df.final2) == 'Case\_Fatality\_Ratio'] <- 'Mortality\_Rate'  statistics <- rbind(df.final, df.final2)  names(statistics)[names(statistics) == "Last\_Update"] <- "Date"  #-----------------------------------------------------------  #Combining datasets  policies$Date <- as.character(policies$Date)  policies$Date <- as.Date(policies$Date, "%Y%m%d")  statistics$Date <- as.Date(statistics$Date)  policies <- policies[with(policies, order(Province\_State, Date)),]  total <- full\_join(statistics, policies)  combined <- total[with(total, order(Province\_State, Date)),]  #Adding daily confirmed and daily deaths columns  combined$Daily\_Confirmed <- combined$ConfirmedCases-lag(combined$ConfirmedCases)  combined$Daily\_Deaths <- combined$ConfirmedDeaths-lag(combined$ConfirmedDeaths)  #Removing unnecessary columns  combined2 <- subset(combined, select = -c(2, 4, 5, 6, 7, 10, 15, 16, 19:22, 46:55, 75:76, 79:88))  #Reordering leftover columns  combined2 <- combined2[, c(1, 2, 55, 53, 3, 56, 54, 4:52)]  #Removing flag columns  combined2 <- subset(combined2, select = -c(16, 19, 22, 25, 28, 31, 34, 38, 52, 55))  #Removing entries before 2020-04-12  combined2 <- combined2 %>% filter(Date>= "2020-04-12")  #List of non-states in the Province\_State column to be removed  not\_states = list("","American Samoa", "Diamond Princess", "District of Columbia", "Grand Princess", "Guam", "Northern Mariana Islands", "Puerto Rico", "Recovered", "Virgin Islands", "Washington DC")  #Removing non-states  combined2 <- combined2[ ! combined2$Province\_State %in% not\_states, ]  #Filling NA's with previous, otherwise filling with next value (per state)  combined3 <- combined2 %>%  group\_by(Province\_State) %>%  fill(names(combined2), .direction = "downup")  #At this point only two columns H4 and H5 had NA's. Those can be replaced by 0's. This can be checked with:  #colnames(combined3)[colSums(is.na(combined3)) > 0]  combined4 <- combined3  combined4[is.na(combined4)] = 0  write.csv(combined4, path\_final\_write, row.names = FALSE) |

# Appendix B

**Preprocessing Spark Code**

|  |
| --- |
| # need this import for functions like "last"  from pyspark.sql.functions import \*  from pyspark.sql import Window  #Oxford cleanup  policies = spark.read.format("csv").option("header", "true").load("/user/hadoop/ox/\*.csv")  policies = policies.withColumnRenamed("RegionName", "Province\_State")  #JHU cleanup  df = spark.read.format("csv").option("header", "true").load("/user/hadoop/jhu1/\*.csv")  df2 = spark.read.format("csv").option("header", "true").load("/user/hadoop/jhu2/\*.csv")  df2 = df2.withColumnRenamed("Total\_Test\_Results", "People\_Tested")  df2 = df2.withColumnRenamed("Case\_Fatality\_Ratio", "Mortality\_Rate")  statistics = df.union(df2)  statistics = statistics.withColumnRenamed("Last\_Update", "Date")  policies.show(n=3)  #Combining datasets  policies = policies.withColumn("Date\_formatted", to\_date(col("Date"), "yyyyMMdd"))  statistics = statistics.withColumn("Date\_formatted", to\_date(col("Date"), "yyyy-MM-dd"))  policies = policies.drop("Date")  statistics = statistics.drop("Date")  policies = policies.withColumnRenamed("Date\_formatted", "Date")  statistics = statistics.withColumnRenamed("Date\_formatted", "Date")  policies.printSchema()  statistics.printSchema()  #policies = policies.withColumn("Date", col("Date").cast('string'))  #policies = policies.withColumn("Date", to\_date(col("Date"), "yyyy-MM-dd"))  #statistics = statistics.withColumn("Date", to\_date(col("Date"), "yyyy-MM-dd"))  #statistics = statistics.select(col("Date"), to\_date(col("Date"), "yyyy-MM-dd").alias("Date"))  #statistics.write.csv("/user/hadoop/final/statistics.csv")  policies = policies.sort("Province\_State", "Date")  policies.write.csv("/user/hadoop/final/policies.csv")  total = statistics.join(policies, ["Province\_State", "Date"], how="full")  combined = total.sort("Province\_State", "Date")  #Adding daily confirmed and daily deaths columns  partition\_cases = Window.partitionBy("Province\_State").orderBy("ConfirmedCases")  partition\_deaths = Window.partitionBy("Province\_State").orderBy("ConfirmedDeaths")  #Getting Daily\_Confirmed and Daily\_Deaths by using the lag function  combined = combined.withColumn("Daily\_Confirmed", combined["ConfirmedCases"]-lag("ConfirmedCases", 1).over(partition\_cases))  combined = combined.withColumn("Daily\_Deaths", combined["ConfirmedDeaths"]-lag("ConfirmedDeaths", 1).over(partition\_cases))  combined.write.csv("/user/hadoop/final/combined.csv")  #Removing unnecessary columns and reordering  columns\_to\_select = ["Province\_State", "Date", "Daily\_Confirmed", "ConfirmedCases", "Recovered", "Daily\_Deaths", "ConfirmedDeaths", "Active", "Incident\_Rate", "People\_Tested", "People\_Hospitalized", "Mortality\_Rate", "Testing\_Rate", "Hospitalization\_Rate", "C1\_School closing", "C1\_Notes", "C2\_Workplace closing", "C2\_Notes", "C3\_Cancel public events", "C3\_Notes", "C4\_Restrictions on gatherings", "C4\_Notes", "C5\_Close public transport", "C5\_Notes", "C6\_Stay at home requirements", "C6\_Notes", "C7\_Restrictions on internal movement", "C7\_Notes", "C8\_International travel controls", "C8\_Notes", "H1\_Public information campaigns", "H1\_Notes", "H2\_Testing policy", "H2\_Notes", "H3\_Contact tracing", "H3\_Notes", "H4\_Emergency investment in healthcare", "H4\_Notes", "H5\_Investment in vaccines", "H5\_Notes", "H6\_Facial Coverings", "H6\_Notes", "H7\_Vaccination policy", "H7\_Notes", "H8\_Protection of elderly people", "H8\_Notes"]  combined2 = combined.select(\*columns\_to\_select)  combined2.printSchema()  combined2.select("Date").show(n=5)  #Removing entries before 2020-04-12  combined2 = combined2.filter(col("Date") >= "2020-04-12").sort("Province\_State", "Date")  #List of non-states in the Province\_State column to be removed  #var notStates = List("", "American Samoa", "Diamond Princess", "District of Columbia", "Grand Princess", "Guam", "Northern Mariana Islands", "Puerto Rico", "Recovered", "Virgin Islands", "Washington DC")  #Removing non-states  combined2.printSchema()  combined2 = combined2.filter(~col("Province\_State").isin(["", "American Samoa", "Diamond Princess", "District of Columbia", "Grand Princess", "Guam", "Northern Mariana Islands", "Puerto Rico", "Recovered", "Virgin Islands", "Washington DC"]))  #combined2 = combined2.drop(col("Province\_State").isin("", "American Samoa", "Diamond Princess", "District of Columbia", "Grand Princess", "Guam", "Northern Mariana Islands", "Puerto Rico", "Recovered", "Virgin Islands", "Washington DC"))  #combined2 = combined2.filter(!combined2.Province\_State.isin("", "American Samoa", "Diamond Princess", "District of Columbia", "Grand Princess", "Guam", "Northern Mariana Islands", "Puerto Rico", "Recovered", "Virgin Islands", "Washington DC"))  combined2.show(n=4)  combined2.write.csv("/user/hadoop/final/combined2.csv")  #------------------------------------------------------------------------------------  import sys  columns\_to\_fill = ["Daily\_Confirmed", "ConfirmedCases", "Recovered", "Daily\_Deaths", "ConfirmedDeaths", "Active", "Incident\_Rate", "People\_Tested", "People\_Hospitalized", "Mortality\_Rate", "Testing\_Rate", "Hospitalization\_Rate", "C1\_School closing", "C1\_Notes", "C2\_Workplace closing", "C2\_Notes", "C3\_Cancel public events", "C3\_Notes", "C4\_Restrictions on gatherings", "C4\_Notes", "C5\_Close public transport", "C5\_Notes", "C6\_Stay at home requirements", "C6\_Notes", "C7\_Restrictions on internal movement", "C7\_Notes", "C8\_International travel controls", "C8\_Notes", "H1\_Public information campaigns", "H1\_Notes", "H2\_Testing policy", "H2\_Notes", "H3\_Contact tracing", "H3\_Notes", "H4\_Emergency investment in healthcare", "H4\_Notes", "H5\_Investment in vaccines", "H5\_Notes", "H6\_Facial Coverings", "H6\_Notes", "H7\_Vaccination policy", "H7\_Notes", "H8\_Protection of elderly people", "H8\_Notes"]  combined\_filled = combined2["Province\_State", "Date"]  combined\_filled.show(n=5)  for col in columns\_to\_fill:      # forward filling NA's - using last known non-NA value  # define the window (grouping by Province\_State)  window = Window.partitionBy('Province\_State')\  .orderBy('Date')\  .rowsBetween(-sys.maxsize, 0)  # define the forward-filled column  filled\_column = last(combined2[col], ignorenulls=True).over(window)  #filled\_column = last(combined2, ignorenulls=True).over(window)  # do the fill  combined2 = combined2.withColumn(col + "\_filled", filled\_column)  combined2 = combined2.drop(col)  #combined3 = filled\_column  combined2.printSchema()  combined2.sort("Province\_State", "Date").show(n=5)  #------------------------------------------------------------------------------------  columns\_to\_fill2 = ["Daily\_Confirmed\_filled", "ConfirmedCases\_filled", "Recovered\_filled", "Daily\_Deaths\_filled", "ConfirmedDeaths\_filled", "Active\_filled", "Incident\_Rate\_filled", "People\_Tested\_filled", "People\_Hospitalized\_filled", "Mortality\_Rate\_filled", "Testing\_Rate\_filled", "Hospitalization\_Rate\_filled", "C1\_School closing\_filled", "C1\_Notes\_filled", "C2\_Workplace closing\_filled", "C2\_Notes\_filled", "C3\_Cancel public events\_filled", "C3\_Notes\_filled", "C4\_Restrictions on gatherings\_filled", "C4\_Notes\_filled", "C5\_Close public transport\_filled", "C5\_Notes\_filled", "C6\_Stay at home requirements\_filled", "C6\_Notes\_filled", "C7\_Restrictions on internal movement\_filled", "C7\_Notes\_filled", "C8\_International travel controls\_filled", "C8\_Notes\_filled", "H1\_Public information campaigns\_filled", "H1\_Notes\_filled", "H2\_Testing policy\_filled", "H2\_Notes\_filled", "H3\_Contact tracing\_filled", "H3\_Notes\_filled", "H4\_Emergency investment in healthcare\_filled", "H4\_Notes\_filled", "H5\_Investment in vaccines\_filled", "H5\_Notes\_filled", "H6\_Facial Coverings\_filled", "H6\_Notes\_filled", "H7\_Vaccination policy\_filled", "H7\_Notes\_filled", "H8\_Protection of elderly people\_filled", "H8\_Notes\_filled"]  for col in columns\_to\_fill2:  # backward filling NA's - using last known non-NA value  # define the window (grouping by Province\_State)  window = Window.partitionBy('Province\_State')\  .orderBy('Date')\  .rowsBetween(0, sys.maxsize)  # define the forward-filled column  filled\_column = first(combined2[col], ignorenulls=True).over(window)  #filled\_column = last(combined2, ignorenulls=True).over(window)  # do the fill  combined2 = combined2.withColumn(col + "\_filled2", filled\_column)  combined2 = combined2.drop(col)  #combined3 = filled\_column  combined2.printSchema()  combined2.sort("Province\_State", "Date").show(n=5)  combined2.write.csv("/user/hadoop/final/combined3.csv")  # at this point only two columns H4 and H5 had NA's. Those can be replaced by 0's  combined4 = combined2.na.fill(0)  combined4.write.csv("/user/hadoop/final/combined\_final.csv") |

# 

# Appendix C

**Query 1 Python Code**

|  |
| --- |
| [  {  '$project': {  'Province\_State': 1,  'Testing\_Rate': 1,  '\_id': 0  }  }, {  '$group': {  '\_id': '$Province\_State',  'Testing': {  '$avg': '$Testing\_Rate'  }  }  }, {  '$sort': {  'Testing': -1  }  }, {  '$limit': 10  }  ] |

**Query 2 Python Code**

|  |
| --- |
| [  {  '$project': {  'Province\_State': 1,  'Mortality\_Rate': 1,  '\_id': 0  }  }, {  '$group': {  '\_id': '$Province\_State',  'Mortality': {  '$avg': '$Mortality\_Rate'  }  }  }, {  '$sort': {  'Mortality': 1  }  }, {  '$limit': 10  }  ] |

**Query 3 Python Code**

|  |
| --- |
| [  {  '$project': {  'Province\_State': 1,  'Recovered': 1,  '\_id': 0,  'recovery\_rate': {  '$divide': [  '$Recovered', '$ConfirmedCases'  ]  }  }  }, {  '$group': {  '\_id': '$Province\_State',  'recovery\_rate': {  '$avg': '$recovery\_rate'  }  }  }, {  '$sort': {  'recovery\_rate': -1  }  }, {  '$limit': 10  }  ] |

**Query 4 Python Code**

|  |
| --- |
| [  {  '$project': {  'Province\_State': 1,  'ConfirmedDeaths': 1,  '\_id': 0  }  }, {  '$group': {  '\_id': '$Province\_State',  'Deaths': {  '$sum': '$ConfirmedDeaths'  }  }  }, {  '$sort': {  'Deaths': -1  }  }, {  '$limit': 10  }  ] |