Database Final Project

Sahba Tashakkori and Jered Dominguez-Trujillo

May 13, 2020

Contents

1 Changes Since Proposal

Due to the fact that the 3 data sources, fatality by age, fatality by sex, and fatality by co-morbidity, were gathered only from New York State, not updated frequently, and not comprehensive, we decided to exclude them. Additionally, we decided to exclude the US Tests dataset provided by the API since it was redundant and only provided the total number of tests done nationally, which was found to be redundant and not useful for a state-level analysis. Instead, we focused on the remaining 4 data sources from the Covid-19 API. They are listed below:

- Covid-19 API
  - **John Hopkins Data Daily Report**: Data for the 2019 Novel Coronavirus Visual sourced from the Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering. Updated daily.
  - **USA Medical Aid Distribution**: Tracks medical aid to US hospitals and facilities, including package weight, cost, delivery date of aid to a location and/or hospital. Updated daily.
  - **Aggregated Facility Capacity County**: Current population, (occupied) hospital beds, (occupied) ICU beds, healthcare staff, age demographics, grouped by county for the United States. Updated daily
  - **United States Cases By State**: For every state: the number of positive and negative cases, each state's score, number of patients in the ICU and on ventilators, and number of patients who have recovered (supplements the JHU Daily Report with additional fields).

2 Introduction

Data on global and United States coronavirus (COVID-19) statistics, medical aid, facilities, and demographics will be gathered and organized into a relational database for the eventual purpose of analysis in order to understand the trends and relationships between demographics, testing, geography, hospital systems/availability, medical aid, and political response to the spread and deadliness of the coronavirus. To do this, several sites and APIs that gather and provide updated timeseries data about the coronavirus were studied and evaluated, including Worldometer, John Hopkins Coronavirus Resource Center, The Covid Tracking Project, and an open source Covid-19 API.

The open source Covid-19 API was chosen, and several data items from the API were utilized to design a database to store time series data relating to COVID-19. The database was then used for some basic analysis and demonstration of queries. The approach taken in choosing which data to source and how it was sourced is presented in Section 3. The process of database design, from conceptual schema, to ER diagram, and finally the translation of the ER diagram to the relational model and implementation in Oracle Cloud DBMS is described in Section 4. Finally, we present and describe some analyses and regression models that we developed using the data extracted from the designed database in Section 5, followed by our conclusions in Section 6.

3 Data Selection

The data sources outlined in Section 2 each had unique data: Worldometer provides updated positive cases and deaths on a rolling 24-hour basis grouped by country, John Hopkins Coronavirus Resource Center provides geographical data updated frequently, The Covid Tracking Project provides data in an easily accessible json format that includes all historical United States positive cases, tests, and deaths data grouped by state, while Covid-19 API provides easily accessibly json data that includes worldwide tests, death, fatality rates, health care system data, and medical aid distribution data that is updated daily and grouped by country, state, city, and county.

After analyzing the data sources available, it was decided that the data would be sourced daily from Covid-19 API since it provides a super-set of the John Hopkins and Worldometer testing and death data as well as information on health care system occupancy and medical aid distribution in the United States.

3.1 Sourcing of Data

This data was accessed with normal GET requests using the Python scripting language. The data was accessed daily, after which it was parsed and re-organized to a csv format that the Oracle Cloud DBMS could easily read, and finally inserted into the DBMS to generate a time series of data. The python script GatherData.py was developed to perform the above task.

Documentation of the various endpoints and data items is included on the Covid-19 API website, but we have included a brief outline of the 4 endpoints (data items) that we used to design our database and perform analysis with:

- Covid-19 API
  - **John Hopkins Data Daily Report**: Data for the 2019 Novel Coronavirus Visual sourced from the Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering. Updated daily.
  - **USA Medical Aid Distribution**: Tracks medical aid to US hospitals and facilities, including package weight, cost, delivery date of aid to a location and/or hospital. Updated daily.
  - **Aggregated Facility Capacity County**: Current population, (occupied) hospital beds, (occupied) ICU beds, healthcare staff, age demographics, grouped by county for the United States. Updated daily.
  - **United States Cases By State**: For every state: the number of positive and negative cases, each state's score, number of patients in the ICU and on ventilators, and number of patients who have recovered (supplements the JHU Daily Report with additional fields).

The data will be gathered daily in order to provide historical data to aide in an analysis of trends over time. This will enable the tracking of tests, positive cases, and deaths in the United States over time. Additionally, data for the United States regarding medical aid distribution, facility capacity, state cases, ICU occupancy, and ventilator usage will be gathered and stored as a time-series for later analysis, modeling, and regression in Section 5.

4 Database Design and Implementation

The database design and implementation for the COVID-19 Timeseries Data is presented as follows:

1. Why a Relational Database (Section 4.1)
2. Approach (Section 4.2)
3. Database Design (Section 4.3)
4. Database Implementation (Section 4.4)
5. Data Insertion and Extraction (Section 4.5)

It should be noted that the Python file used to collect and prepare the data for database insertion is included as GatherData.py, the SQL files used to create the database and populate the initial STATE table are included as createDB.sql and PopulateStateTable.sql, the SQL file to extract data for analysis using queries is included as Queries.sql, and the Python notebook used to analyze the data and generate the plots seen in Section 5 is included as Analysis.ipynb.

4.1 Why a Relational Database

The data that we worked with fits into a single machine. Therefore, one of the main motivations behind the NoSQL movement, which was the the ability to distribute relational databases, does not apply to our scenario. In addition, we will not have the "impedance mismatch problem" between our storage and representation formats that NoSQL databases aim to solve [1]. Hence we do not need to use NoSQL solutions, such as document or column based databases, to ease our representation and operations. Moreover, the need for timestamps can be easily solved by adding a timestamp/date column to each entry that keeps track of when the data was recorded and entered. So using a specialized time series database is simply not necessary and results in pointless complications in our approach. We also do not have a data ingestion problem. We will gather the data once per day at a specific time by writing a script to download the data from the website, parse the data, and insert the new data into the database, as described in Section 4.5. For these reasons, a relational database design was chosen to represent the data relationships and to store the COVID-19 data.

4.2 Approach

Since we worked with 4 different data sources from the Covid-19 API, we approached this problem by first representing each data source as an entity, all related to a parent STATE entity, as seen in Figure 1. Since the data provided from the API was largely data from the United States of America, and since the USA has the largest COVID-19 outbreak in the world. it was decided to focus solely on US data and decompose by State. Since the intent was to represent time series data, it was decided that an additional column would be added to each data source that indicated the access/record date. This would allow data to be stored over time in such a way that test data, case data, facility data, and medical aid data could be observed as it changed over time, and the state of the COVID-19 outbreak in the United States on any day could be extracted.

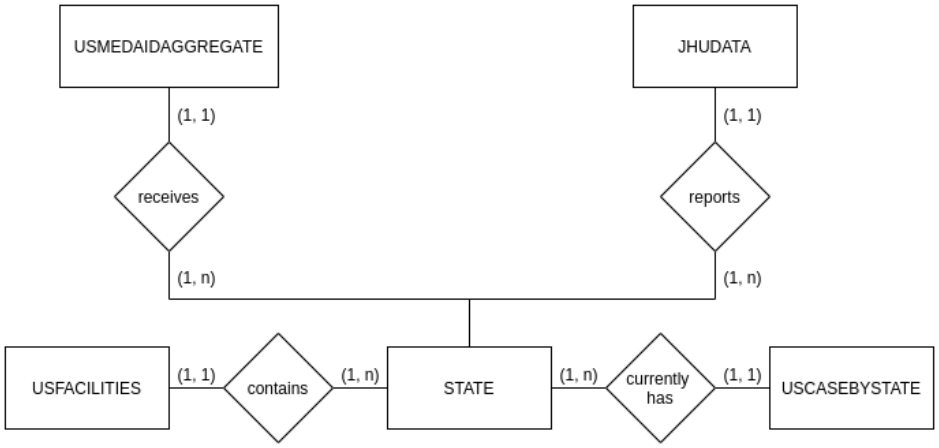


Figure 1: Initial Entity-Relationship Diagram to Model our Data Sources and Relational Database Design

4.3 Database Design

The database design began with a simple ER diagram, where each data source was assigned a relationship to a parent STATE entity, as shown in Figure 1. As will be seen later on, in the actual implementation in section 4.4, the data was easily translated to the mostly Third Normal Form (3NF), since the only functional dependency present in the relations was that of the primary key (State and DateRecorded) determining all other columns (with the exception of the JHUDATA, USFACILITIES, USALLBEDOCCUPANCY, and USICUBEDOCCUPANCY, which are 2NF because of the CountyName and Country keys). Again, note that the Total column in USCASEBYSTATE is defined not functionally dependent on the positive and negative columns due to the existence of inconclusive tests and/or data reporting anomalies. Also, to reduce the number of NULL values present in the database, the United States Cases By State (USCASEBYSTATE) data source was broken up into several entities, such as recovery, ventilator, hospitalized, and ICU information (with each of those entities only having an entry for each state that had at least one non-NULL entry for the day). A similar decomposition was performed for the Aggregated Facility Capacity Count (USFACILITIES) data source into separate ICU bed information and overall bed information. This resulted in a revised ER diagram, seen in Figure 2. Finally, this ER diagram was translated to the relational model, as seen in Figure 3. As later discussed in Section 4.4, most of these tables are in fact in Third Normal Form (3NF), such that natural joins result in no spurious tuples and that all of the columns are functionally dependent on solely the primary key, while the remaining tables mentioned are 2NF and not normalized further since, for our purposes, it would have added unnecessary complications.

4.3.1 Conceptual Schema

The conceptual schema for the COVID-19 Database is presented as follows:

- Entities and Attributes
- Relationships
- Final ER Conceptual Schema Diagram (Figure 2)
- Explicit Integrity Constraints

As is typical, it should be noted that the keys of the entities have been indicated using **bold font and a regular underline**.

Entities and Attributes

The entities defined for this databse are:

- STATE
- JHUDATA
- USMEDAIDAGGREGATE
- USFACILITIES
- USICUBEDOCCUPANCY
- USALLBEDOCCUPANCY
- USCASEBYSTATE
- RECOVERED
- VENTILATOR
- HOSPITALIZED

A more detailed description of each entity follows:

STATE: One of the 50 political entities, 5 territories, and the District of Columbia that together make up the U.S. Additionally, the Diamond Princess cruise ship, the Grand Princess cruise ship, and an entry for Recovered patients without a state are included, for a total of 59 entries. See explicit integrity constrains for more information.

Some example instances are: New Mexico, New York, California, Georgia.

Attributes:	**StateAbbreviation**	(SSPF)
	StateName	(SSPF)

JHUDATA: John Hopkins Daily COVID-19 Data.

An example instance is: JHU Data from Albuquerque, New Mexico for May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	Country	(SSPF)
	State	(SSPF)
	City	(SSPF)
	Confirmed	(SSPF)
	Deaths	(SSPF)
	Recovered	(SSPF)
	Active	(SSPF)
	Latitude	(SSPF)
	Longitude	(SSPF)

USCASEBYSTATE: Daily COVID-19 Data broken down by State.

An example instance is: Number of positive, negative, and total tests performed in New Mexico for May 10, 2020, in addition to the number of deaths and the quality of the data for the day.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	Positive	(SSPF)
	Negative	(SSPF)
	Death	(SSPF)
	Total	(SSPF)
	DataQualityGrade	(SSPF)
	DataGrade	(SSPF)

Notes

1. DataQualityGrade: The grade on an A-F scale, of the quality of data provided by the state for a specific day.
2. DataGrade: The grade on an A-F scale of the data (i.e. how severe is the outbreak in the state) of a state for a specific day.

HOSPITALIZED: Number of hospitalizations per state on a daily basis.

An example instance is: Number of hospitalized patients currently and cumulatively (since the beginning of the pandemic) in New Mexico for May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	HospitalizedCurrently	(SSPF)
	HospitalizedCumulative	(SSPF)

ICU: Number of patients in the ICU per state on a daily basis.

An example instance is: Number of ICU patients currently and cumulatively (since the beginning of the pandemic) in New Mexico for May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	InICUCurrently	(SSPF)
	InICUCumulative	(SSPF)

VENTIALTOR: Number of patients on a ventilator per state on a daily basis.

An example instance is: Number of patients currently on a ventilator and the number of cumulative patients who have required a ventilator since the beginning of the pandemic in New Mexico for May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	OnVentilatorCurrently	(SSPF)
	OnVentilatorCumulative	(SSPF)

RECOVERED: Number of recovered COVID-19 patients per state on a daily basis.

An example instance is: Number of recovered patients since the beginning of the pandemic in New Mexico for May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	Recovered	(SSPF)

USFACILITIES: Population, age distribution, and bed/ICU availability on a per county granularity.

An example instance is: Population, demographics, and number of beds in Bernalillo County, New Mexico on May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	CountyName	(SSPF)
	Population	(SSPF)
	Population20Plus	(SSPF)
	Population65Plus	(SSPF)
	StaffedAllBeds	(SSPF)
	StaffedICUBeds	(SSPF)
	LicensedAllBeds	(SSPF)

USALLBEDOCCUPANCY: Current occupancy of hospital beds on a per country granularity.

An example instance is: Number between 0 and 1 indicating the level of hospital beds that are occupied in Bernalillo County on May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	CountyName	(SSPF)
	AllBedOccupancyRate	(SSPF)

USICUBEDOCCUPANCY: Current occupancy of ICU beds/units on a per country granularity.

An example instance is: Number between 0 and 1 indicating the level of ICU beds/units that are occupied in Bernalillo County on May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	CountyName	(SSPF)
	ICUBedOccupancyRate	(SSPF)

USMEDAIDAGGREGATE: Cumulative amount of medical aid sent to a state since the start of the pandemic. Includes number of packages, cost of material, and weight of received packages.

An example instance is: Cumulative number of packages, the cost of the packages, and the weight of the packages received by New Mexico up to May 10, 2020.

Attributes:	**DateRecorded (Month-Day-Year)**	(CSPF)
	State	(SSPF)
	Deliveries	(SSPF)
	Cost	(SSPF)
	Weight	(SSPF)

Relationships

The relationships in this schema are listed and described below.

reports A state performs COVID-19 tests through John Hopkins site, each of which may be positive or negative. This relationship keeps track of the daily tests in the United States and their associated data.

Attributes: None

receives A state receives medical aid from industry or the federal government. This relationship keeps track of the aid received by states in terms of the cost of the aid, the number of received packages, and the weight of the received packages.

Attributes: None

contains A state contains facilities, such as hospitals, beds, and ICU units. This relationship keeps track of the number of these available to each state, and the current occupancy level of the available facilities.

Attributes: None

provides A facility provides a certain number of ICU units, and each facility has a certain occupancy rate of those units. This relationship keeps track of the occupancy rate and availability of those ICU units.

Attributes: None

supplies A facility also supplies a certain number of staffed beds. This relationship keeps track of how many beds a facility supplies and how many beds a facility is currently using (as a rate).

Attributes: None

currentlyHas Each state currently has a certain number of COVID-19 cases. This relationship keeps track of those cases by state, in addition to allowing the information from child relationship and entities, such as recovered patients, patients in the ICU, patients on ventilators, and hospitalized patients, to propagate upwards and be associated with particular states.

Attributes: None

patients Each set of cases has a number of recovered patients, which this relationship keeps track of daily.

Attributes: None

connectTo Each set of cases in a state has a number of patients connected to ventilators. This relationship keeps track of that statistic at the statewide level on a daily basis.

Attributes: None

occupied Each set of cases in a state has a count of the number of patients that are currently occupying ICU units. This relationship keeps track of and updates that statistic daily.

Attributes: None

checkedIn Each set of cases in a state has a current count of the number of hospitalized COVID-19 patients, and this relationship tracks that daily.

Attributes: None

Explicit Integrity Constraints

Some examples of integrity constrains in our COVID-19 database follow:

1. In the STATE entity, there should be exactly 59 entries (50 states, 5 territories and DC, and the Diamond Princess, Grand Princess, and Recovered) in order to accommodate the data from the API.
2. The USCASEBYSTATE entity should have 56 new entries for each day it is updated (one for each of the 50 states, 5 territories and DC). For example, if data collection has been occurring for 6 days, there should be exactly 336 entries.

Final ER Conceptual Schema Diagram

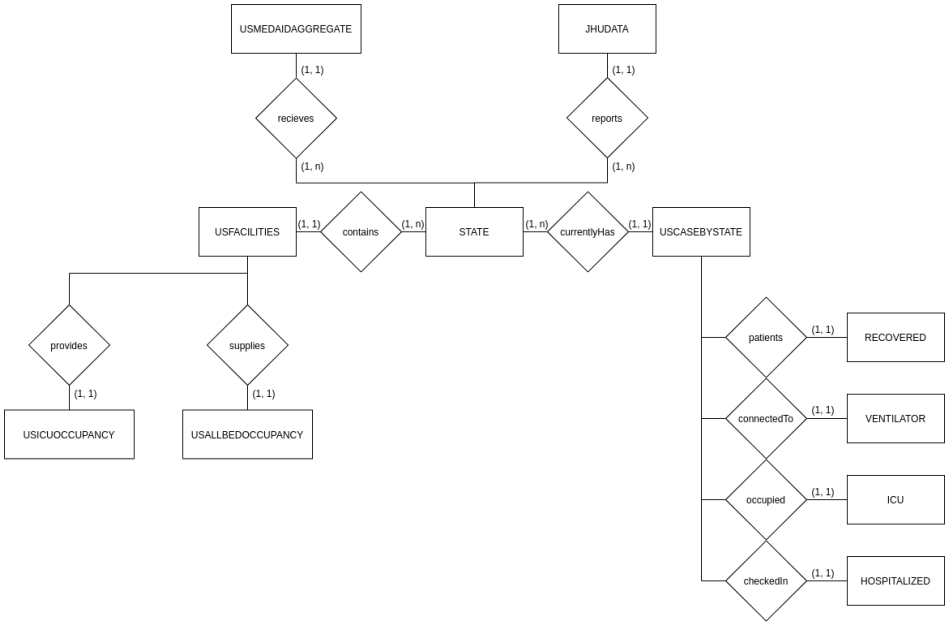


Figure 2: Basic Entity-Relationship Diagram to Model our Data Sources and Relational Database Design

4.4 Database Implementation

The database was constructed by translating the final ER Diagram in Figure 2 to the relational model, as seen on the next page in Figure 3. A SQL script was then written (CreateDB.sql) to implement the schema in the Oracle Cloud DBMS, and the database was then used to collect data to provide the ability to perform useful queries that aided in the analysis presented in Section 5.

Note that every table in the relational database shown in Figure 3 is in fact in 3NF, except for the JHUDATA, USFACILITIES (and its children USALLBEDOCCUPANCY and USICUBEDOCCUPANCY) tables which are 2NF. Note that the JHUDATA and USFACILITIES tables were not further normalized or decomposed as it seemed unnecessary since we are solely focusing on US data and additional CITY, COUNTY and/or COUNTRY tables would have been unneeded and unnecessarily complicated for our purposes. For the STATE table, the StateName is functionally dependent solely on the StateAbbreviation private key, while for the remaining tables, their columns are functionally dependent solely on the primary key of DateRecorded and State. Lastly, note that the Total columns in some of the tables are not functionally dependent on the positive and negative columns, since due to anomalies with reporting, inconclusive tests, or un-reported tests, the value in the Total column may be different than the sum of the positive and negative reported test columns and therefore is important to include and not normalize away.

Also note that DateRecorded was an added column to the raw data to allow for the storage of a snapshot of the state of the COVID-19 pandemic in the United States on each day. This provided the time series capability of the database, and resulted in the peculiar property that each tuple once entered, is not modified such that the database only continues to grow each day as new entries for the day are entered (i.e. only INSERT operations and no UPDATE operations).

Figure 3: COVID-19 Relational Schema

4.5 Data Insertion and Extraction for Analysis

The data was collected between 5pm and 7pm MST everyday from May 6, 2020 until May 13, 2020 using the Covid-19 API and a python script (GatherData.py). This process involved posting a GET request to the 5 endpoints described in Section 3.1 and obtaining the data in JSON format.

Since some of the data included had irrelevant and extraneous columns, the python script first removed those extra columns and converted the data to CSV format to begin prepping for upload to our database instance on Oracle Cloud. Next, since the data needed to be broken down into a file for each of the 11 tables (excluding that STATE table), the python script organized the data into 11 files, taking care that the columns in each CSV file matched the columns of the corresponding table in the instance of the database.

This allowed the users to run the python script once every evening to collect and organize data, and subsequently upload the data into Oracle Cloud. Backups of the raw data obtained from the COVID-19 API endpoints, in addition to backups of the processed data, were kept for each day the script was run. This process and the results of the analyses from the data entered and extracted from the database is covered in Section 5. An overview of this process can be found in Figures 4 - 5.

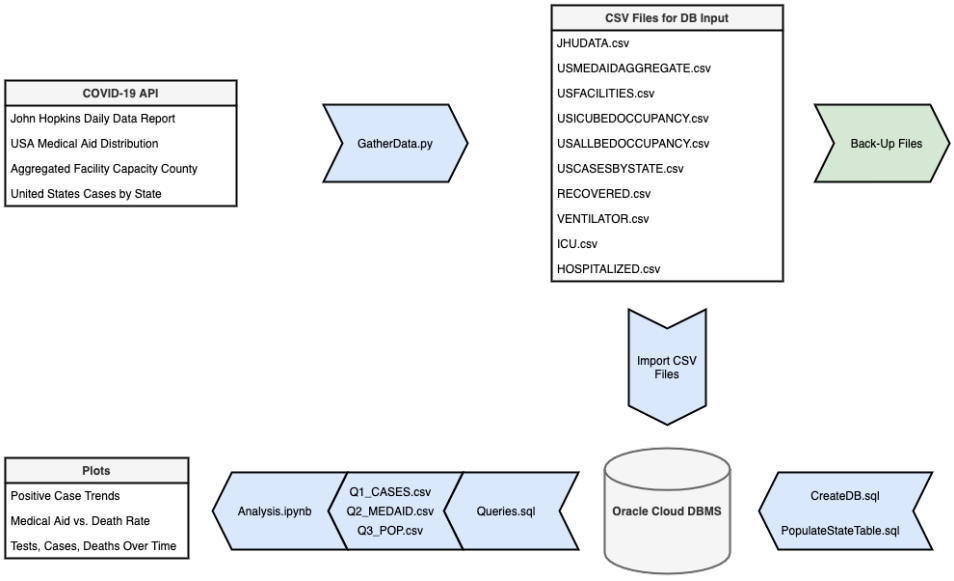


Figure 4: Data Insertion and Extraction Workflow Overview

5 Data Analysis

We performed a number of analyses to demonstrate the viability of our design and collected data for the kind of analyses that both experts and enthusiasts might be interested in performing. We mainly used SQL queries and Python's data science toolset (Pandas[2], NumPy[4], ScikitLearn[3]) to retrieve, prepare, and analyze our data. The following sections provides details on our approach in preparing the data and performing the analyses.

5.1 Data Preparation

We divided the large tuples of raw data which we retrieved from our sources into multiple tables (Figure 5). In addition to resulting in a more logical representation, this approach resulted in far fewer NULL values in the table. The Python script to retrieve and divide the data resides in the *GatherData.py* file.

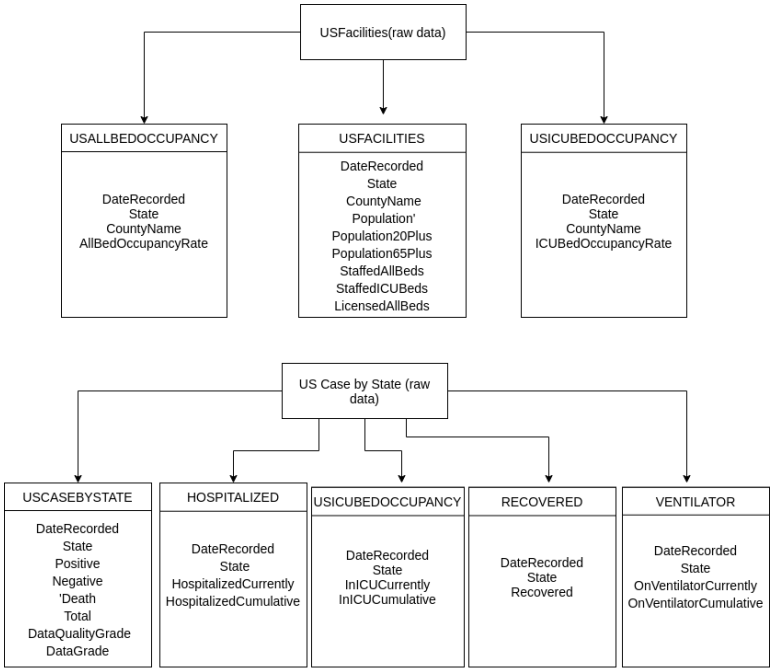


Figure 5: Breaking down the raw data into multiple tables. Note that the arrows simply represent the flow of data and do not connote any database design related concept

We explored the following items in the data:

- Trends in tests, positive and negative cases, and deaths at the State Level
- Correlation of tests with medical aid at the state level
- Hospital resources as a predictor of death rate

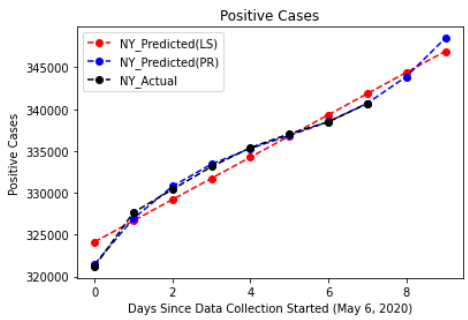
In order to perform these analyses, we used simple SQL queries against our database to retrieve the data and simple python scripts to analyze the data and visualize the results. The queries used to retrieve the historic data from our database can be found in the *Queries* folder and the Python scripts for analyzing and visualizing the data can be found in the *analysis.ipynb* Jupyter Notebook.

5.2 Tests, Cases, and Deaths at the State Level

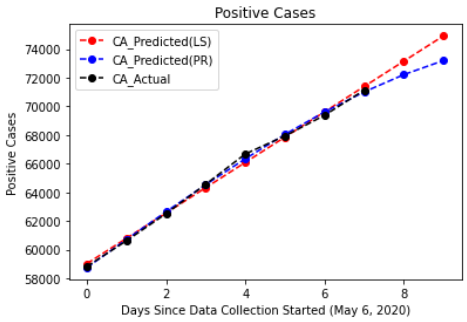
Figure 6 shows the number of positive cases in four states and the predictions projected two days into the feature using regression analysis.

5.2.1 Approach

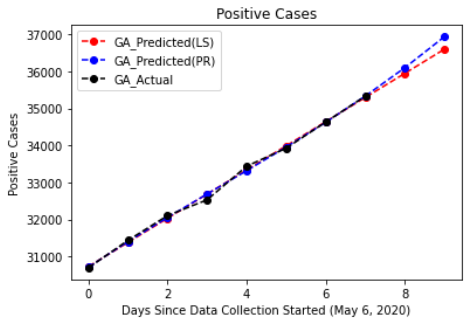
We used the 'USTESTBYSTATES' table to retrieve historical trends for the states. The result of the query was exported to a *csv* file and imported into the python script for plotting and regression analysis.



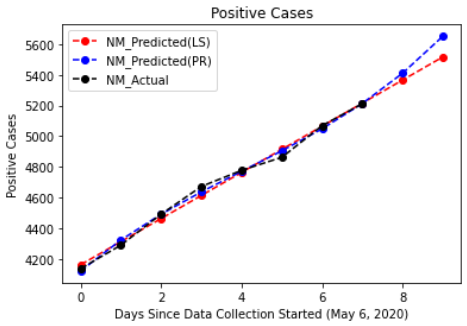
(a) Positive cases trends for NY



(b) Positive cases trends for CA



(c) Positive cases trends for GA



(d) Positive cases trends for NM

Figure 6: The number of positive cases for CA, GA, NY, NM and predictions for two days into the future using linear(LS) and polynomial regression (PR3)

The SQL command used to obtain this data is presented below:

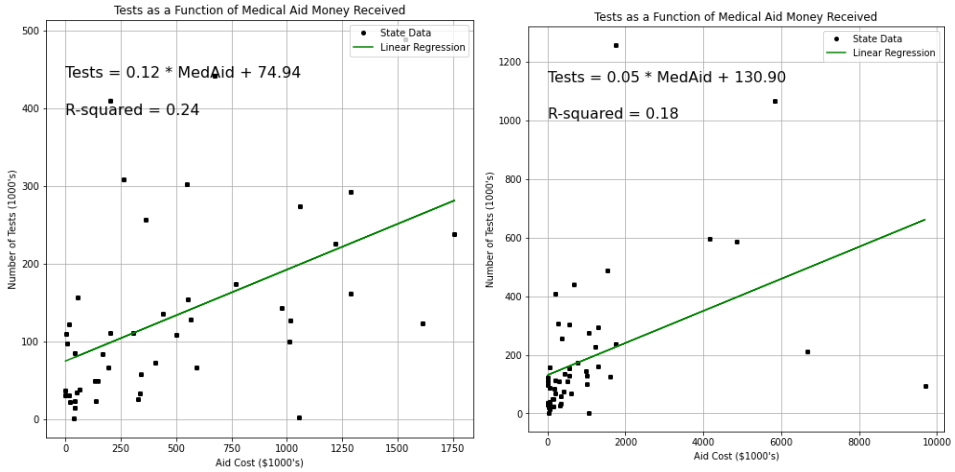
```
1  SELECT DateRecorded, State, Positive, Negative, Total, Death
2  FROM USCASEBYSTATE
3  ORDER BY State;
```

5.3 Correlation of Tests with Medical Aid at the State Level

We looked at the historical test and medical aid data in our database to look for the relationship between the amount of medical aid a state receives and the number of tests the state performs.

5.3.1 Approach

We joined two tables 'USCASEBYSTATE' and 'USMEDAIDAGGREGATE', on the timestamp and state attributes and returned the total monetary value of the aid received and the number of tests performed.



(a) Without outliers.

(b) With outliers.

Figure 7: Number of tests conducted vs the amount of medical aid received by states (for all 50 states, 5 territories, and DC) with a line fit through the data using least squares.

The SQL command used to obtain this data is presented below:

```
1  SELECT u.DateRecorded, u.State, u.Positive, u.Negative, u.
   Total AS test_total, u.Death, ma.Deliveries, ma.Cost, ma.
   Weight
2  FROM USCASEBYSTATE u, USMEDAIDAGGREGATE ma
3  WHERE u.State = ma.State AND u.DateRecorded = ma.
   DateRecorded
4  ORDER BY u.State, u.DateRecorded;
```

5.4 Hospital Resources as a Predictor of Cases and Deaths

In order to demonstrate the viability of our database for more ambitious analyses, we decided to look at the occupancy rate of ICUs and the number of deaths per capita.

5.4.1 Approach

We performed a 3-way join on JHUDATA, USICUBEDOCCUPANCY, and US-FACILITIES and performed some aggregate operations to obtain the average ICU occupancy rate and total number of fatalities and the population for each state. We normalized the total number of fatalities and plotted them as shown in Figure 8.

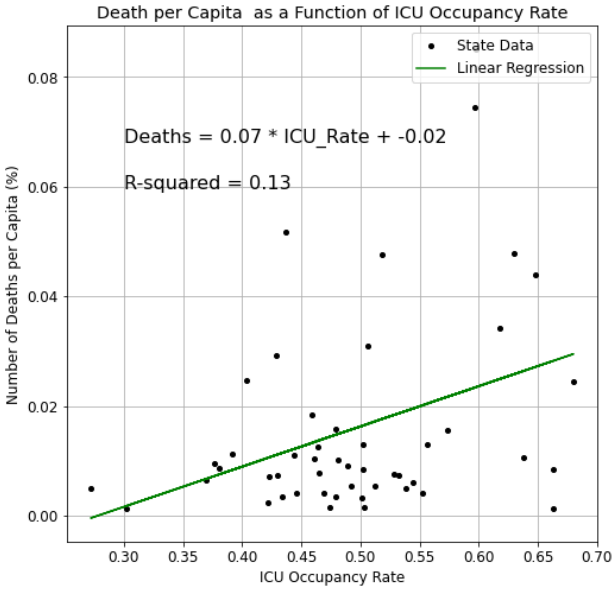


Figure 8: The rate of death per capita vs the ICU occupancy rate (of all 50 states, 5 territories, and DC) with a line fit through the data using least squares

Obviously, according to Figure 8, there does not seem to be a strong correlation between ICU occupancy rate (as reported) and the number of deaths per capita. However, there are always issues of whether the reported data is accurate, and there are many other variables that could influence the death rate in specific regions.

The SQL command used to obtain this data is presented below:

```
SELECT f.State, f.DateRecorded, total_pop, icu_avg, j.
death_tot
FROM(
    SELECT f.State, f.DateRecorded, sum(Population) AS
total_pop
    FROM USFACILITIES f
    WHERE  DateRecorded=to_date('05/13/2020','MM/DD/YYYY')
    GROUP BY State, DateRecorded
    ) f ,
    (SELECT o.State, o.DateRecorded, round(avg(
ICUBedOccupancyRate),3) AS     icu_avg
    FROM USICUBEDOCCUPANCY o
    WHERE o.DateRecorded=to_date('05/13/2020','MM/DD/YYYY')
    GROUP BY o.State, DateRecorded
    ) o,
    (SELECT sum(Deaths) AS death_tot, j.State, j.
DateRecorded
    FROM JHUDATA j
    WHERE DateRecorded=to_date('05/13/2020','MM/DD/YYYY')
    GROUP BY State, DateRecorded) j
WHERE f.State=o.State AND j.State=o.State;
```

6 Conclusions

In this work we designed a database for keeping a history of COVID-19 data. We demonstrated that our design is able to incorporate and store data from multiple sources and present them in a logical and practical fashion. Adding the timestamps and using natural identifiers, such as the state names, for the records allowed us to easily combine the data gathered from different sources and use them in our sample analyses. These sample analyses showed that our design and the data we collect allow for statistical studies to unveil COVID-19 related trends and correlations. If in order to perform a new analysis, it becomes necessary to incorporate new sources into the database, the new data can be easily integrated into our design to be used in conjunction with the existing data.

As we planned and executed different phases of this project, we realized that the most involved and sensitive part of the project was designing the database. In other words, once a good database design was achieved and the data was stored properly, retrieving the data and performing analyses on it was fairly straightforward.

A common question regarding COVID-19 is that whether or not the temperature has a meaningful impact on the spread of the disease. Hence an interesting possible extension for this work is finding a reliable source of weather information and combining that with the existing mortality and disease progression data. A careful researcher would be able to use the data to potentially study the impact of temperature on the spread of the virus while normalizing for other factors.

References

[1] Fowler, M. Introduction to nosql (webinar), 2012. [Online; accessed 20-April-2020].

[2] McKinney, W. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 51 – 56.

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[4] van der Walt, S., Colbert, S. C., and Varoquaux, G. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering 13*, 2 (March 2011), 22–30.

A Other Analyses and Visualizations

Lastly, we provide a few visualizations of the progression of the COVID-19 pandemic from May 6, 2020 - May 13, 2020 with the charts below. The data required for the visualizations below was obtained with the same SQL command as was used in section 5.2.

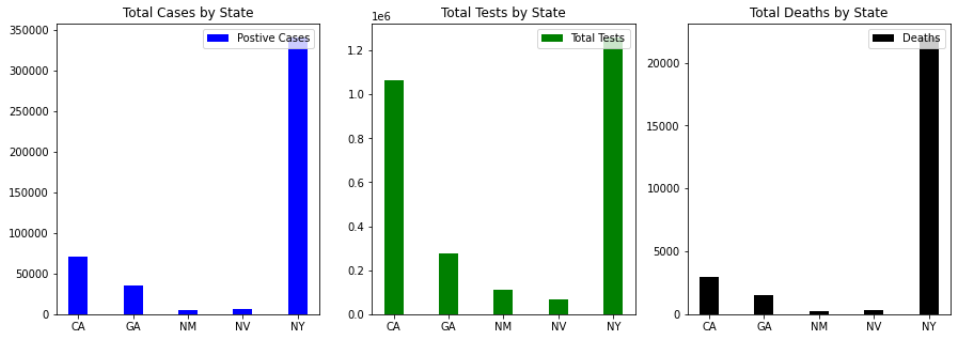


Figure 9: Total Cases and Deaths in California, Georgia, New Mexico, Nevada, and New York, as of May 13, 2020.

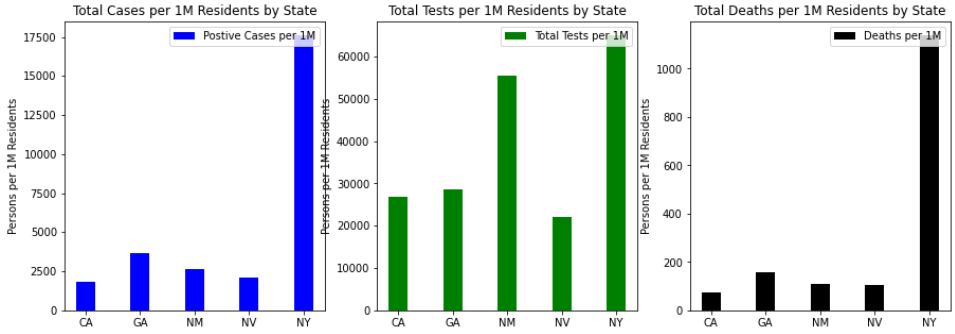


Figure 10: Cases and Deaths per 1 million residents in California, Georgia, New Mexico, Nevada, and New York, as of May 13, 2020.

As can be seen in Figure 9, by the raw values, New York is performing the most tests, has the most cases, and has the most deaths by a large margin (California has performed a similar number of tests). However, once normalized by population (Figure 10), we see that New Mexico is also performing a substantial number of tests per capita, while Georgia has the second most cases and deaths per capita of the 5 states, behind New York.

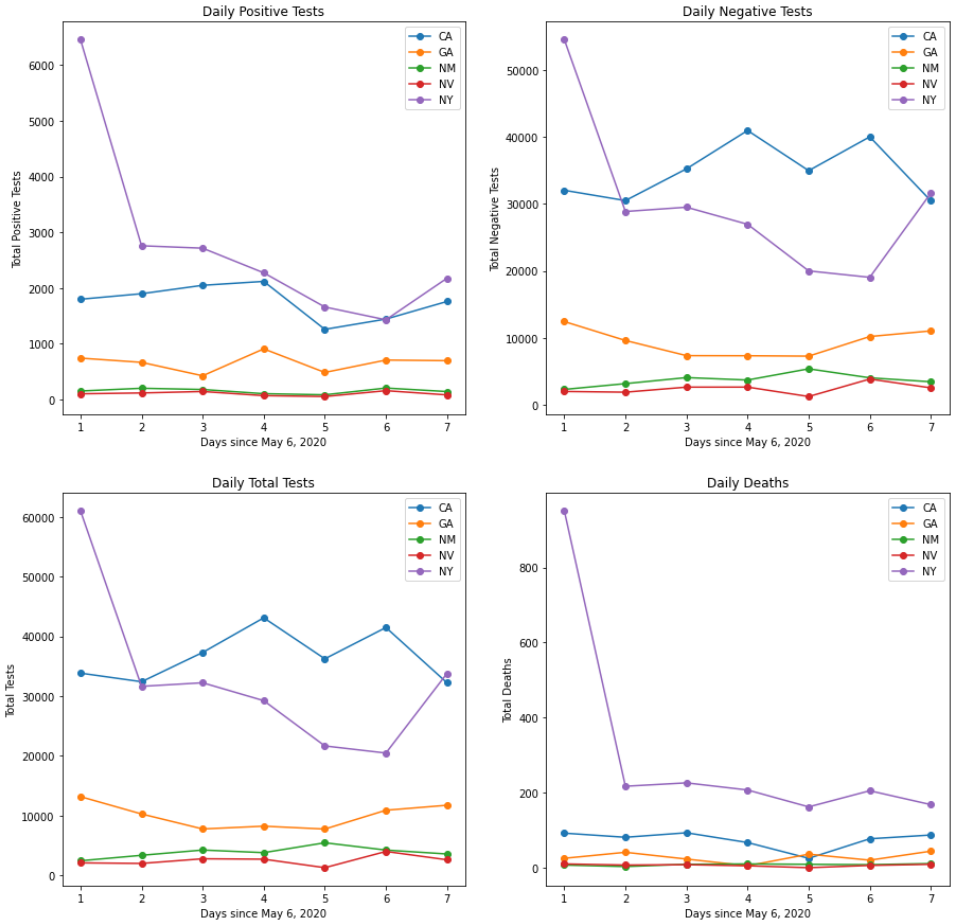


Figure 11: Daily Tests from May 6 - May 13 in California, Georgia, New Mexico, Nevada, and New York.

As can be seen in Figure 11, it is apparent that New York is on the down slope, as their number of positive tests and deaths continues to decrease. However, due to only gathering data for a week and knowing that the data reports have a weekly oscillation (typically peaks on Tuesday and is low Saturday-Monday), it is hard to draw many conclusions from only a week of data that we have been able to gather.

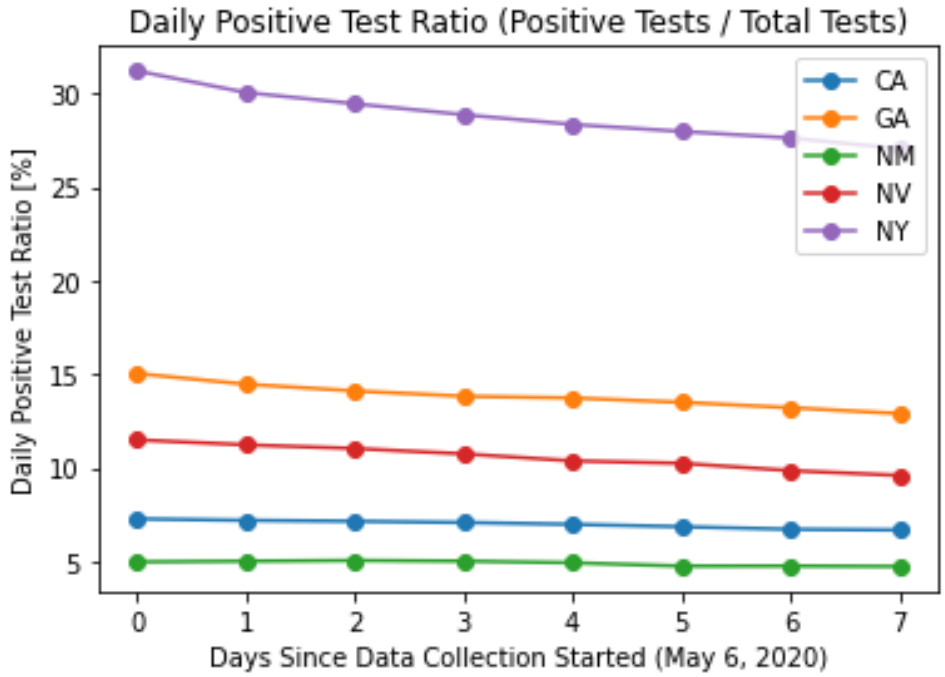


Figure 12: Daily Positive Test Ratio from May 6 - May 13 in California, Georgia, New Mexico, Nevada, and New York.

As can be seen in Figure 12, as the volume of testing continues to increase and more people continue to stay at home, the daily positive testing ratio continues to decrease across all 5 states, with New York having the worst ratio and New Mexico having the best ratio.

The code used to generate these plots can be found in Analysis.ipynb.