

# Covid-19 AWS Data Integration Project

## Problem Setting:

In recent times, especially after covid pandemic the global infrastructure has faced a lot of difficulties with hospitals, beds, medical supplies, staff. Thus, it has become more important now than ever for the government and the health societies to an unfortunate situation. But, this involves a large amount of data and might not be handled by the on-premises Datawarehouse, this cloud is the best solution to store and analyze data.

## Problem Definition:

The large amount of data created for the COVID-cases and available medical supplies, needs to be assessed. This must be done in a computationally efficient manner. To handle this data, we must look for scalable and robust solutions. Our proposal involves using cloud-based warehouses to store, manipulate and analyze the COVID data, which is at a daily level. We intend to analyze trends and interpret them in our project.

## Dataset:

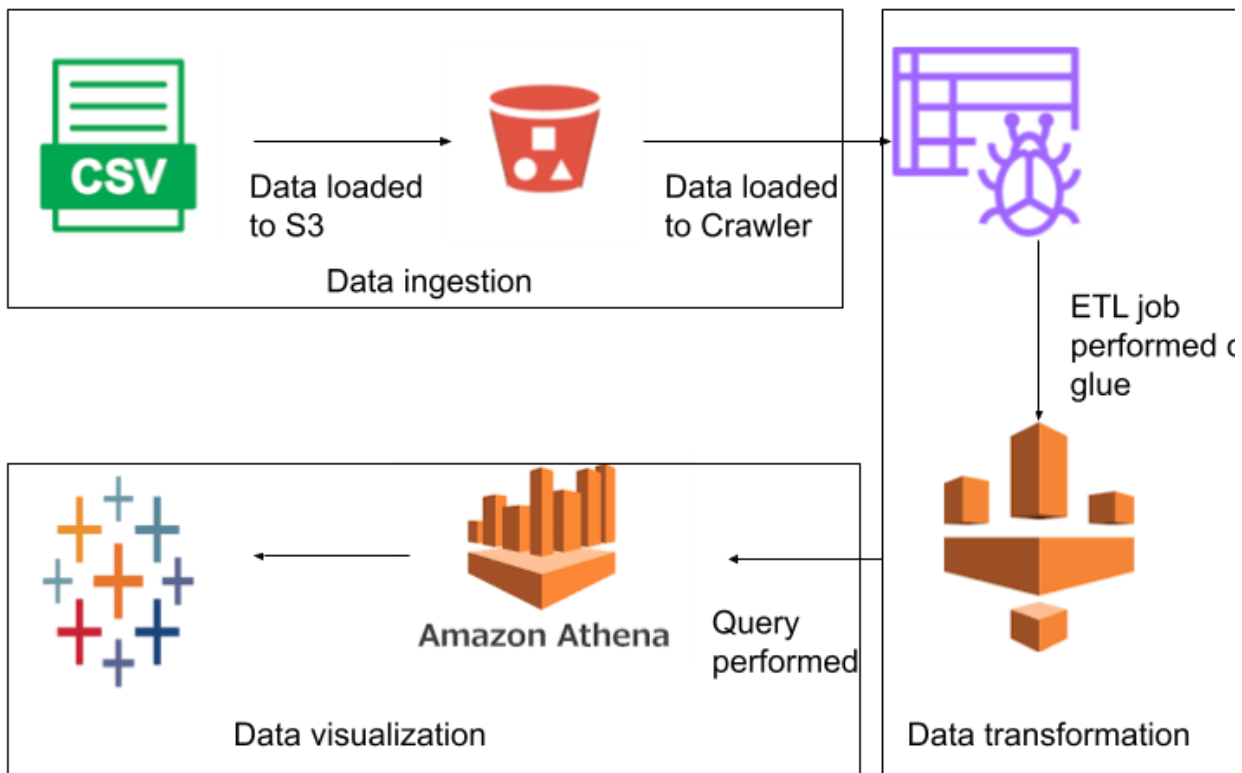
Our dataset consists of 10 tables, that include - vaccination distribution, COVID cases (confirmed cases, deaths), New York state and county details, NY daily testing details, hospital beds data, a few lookup country and state tables that contain abbreviation's.

Dataset Link: [COVID-Dataset](#)

## Analysis Goal:

1. Analyze the trends in Covid Confirmed cases to Hospital bed required, including patterns related to time of day, day of the week, and month.
2. To analyze the rate of spread of such infectious diseases within a confined region (counties) on a day-to-day basis. Hence observing percentage of population affected in counties.
3. Get a gist of an individual's reaction by analyzing Foot traffic in public venues like airports, gyms, and grocery stores.
4. Impact of number of cases on Hospital beds in region and impact of population density on hospital beds. Thereafter concluding how Health societies should do in such future events.
5. Analyzing trend in covid cases to that of vaccination. Also getting trend on the vaccination done in USA

## Data Architecture:



### Amazon S3:

(Simple Storage Service) is a cloud storage service provided by Amazon Web Services (AWS). An S3 bucket is a container for storing and organizing data objects, such as files and images, in the cloud. It allows users to store and retrieve any amount of data from anywhere on the web. S3 buckets are highly scalable, reliable, and can be configured with various access controls to manage permissions for different users or applications. They are commonly used for backup, data archiving, website hosting, and as a storage backend for various applications in the cloud.

### AWS Crawler:

In AWS, a crawler is a component of AWS Glue, a fully managed extract, transform, and load (ETL) service. The crawler automatically discovers and extracts metadata from various data sources such as databases, data lakes, or flat files. It helps in understanding the structure of the data and creates metadata tables, which can then be used for data processing and analysis. Essentially, a crawler in AWS automates the process of cataloging and organizing data from diverse sources, making it easier to work with in analytics and data processing tasks.

### AWS Glue:

AWS Glue is a fully managed Extract, Transform, and Load (ETL) service by Amazon Web Services. It automates the process of preparing and loading data for analysis. Glue offers data cataloging, ETL job creation without manual coding, a serverless Apache Spark environment, job scheduling, and tools for data preparation and cleaning. It simplifies data processing workflows and provides a scalable, serverless platform for managing and transforming data in the cloud.

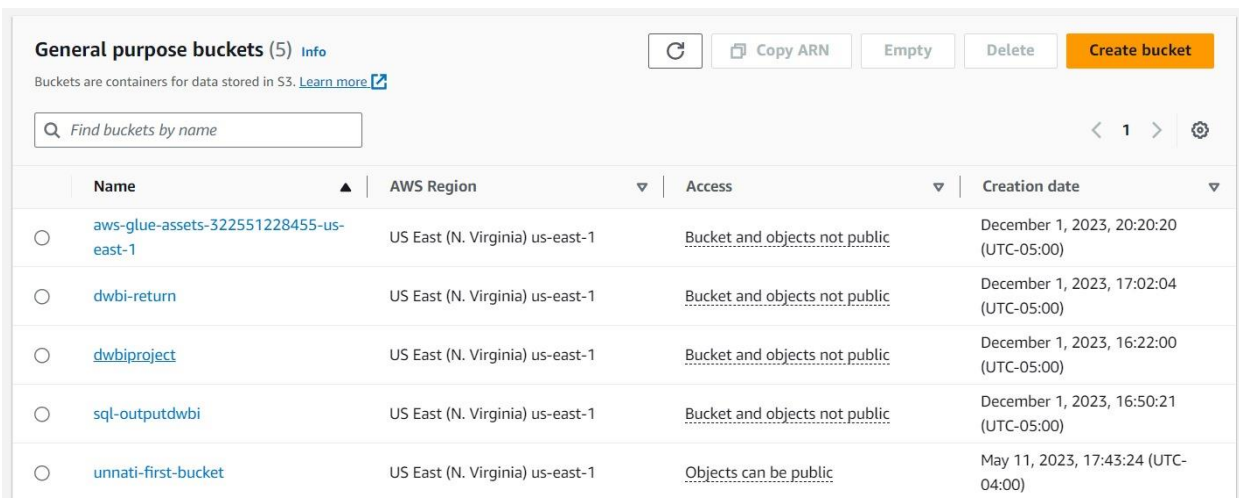
## AWS Athena:

AWS Athena is a serverless query service provided by Amazon Web Services. In short, Athena allows you to analyze data stored in Amazon S3 using standard SQL queries. It eliminates the need for managing infrastructure or databases, as you can directly query data in S3 without the need to load it into a separate database. Athena is particularly useful for ad-hoc analysis, reporting, and extracting insights from large datasets in a cost-effective and scalable manner.

At the end, Tableau will be used to visualize data and create dashboards.

## Data Ingestion:

- Prepared structured data in CSV format.
- Created an S3 bucket named "dwbiproject".
- Inside the bucket, made individual folders for each of the 10 tables.
- Stored respective CSV files for each table in its folder.
- Achieved organized storage for ease of access and management.



The screenshot shows the AWS S3 console interface for 'General purpose buckets (5)'. At the top, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. Below these is a search bar labeled 'Find buckets by name'. The main content is a table with columns: Name, AWS Region, Access, and Creation date. The table lists five buckets, all in the 'US East (N. Virginia) us-east-1' region. The first four buckets have 'Bucket and objects not public' access, while the last one, 'unnati-first-bucket', has 'Objects can be public' access.

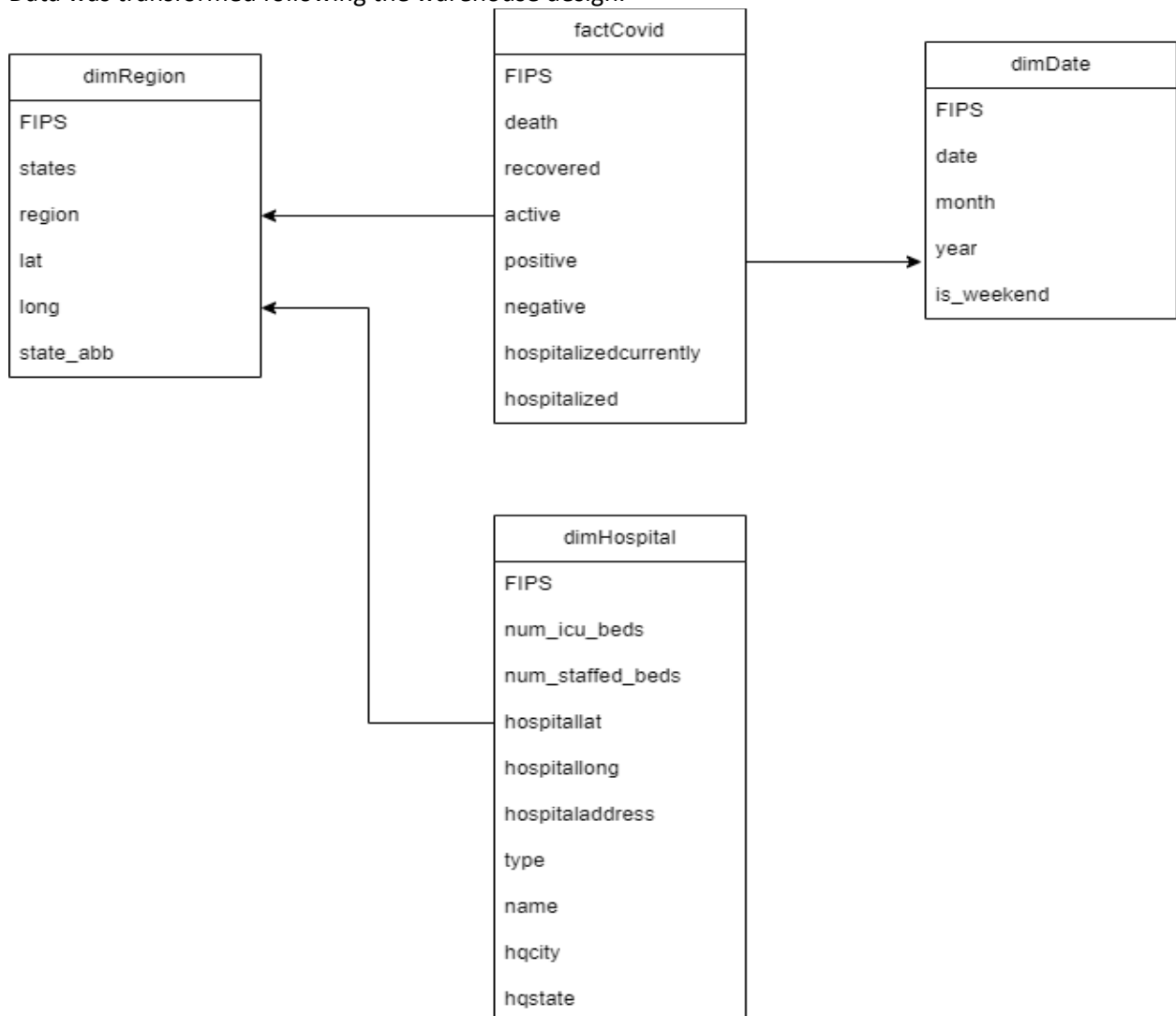
	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	<a href="#">aws-glue-assets-322551228455-us-east-1</a>	US East (N. Virginia) us-east-1	<a href="#">Bucket and objects not public</a>	December 1, 2023, 20:20:20 (UTC-05:00)
<input type="radio"/>	<a href="#">dwbi-return</a>	US East (N. Virginia) us-east-1	<a href="#">Bucket and objects not public</a>	December 1, 2023, 17:02:04 (UTC-05:00)
<input type="radio"/>	<a href="#">dwbiproject</a>	US East (N. Virginia) us-east-1	<a href="#">Bucket and objects not public</a>	December 1, 2023, 16:22:00 (UTC-05:00)
<input type="radio"/>	<a href="#">sql-outputdwbi</a>	US East (N. Virginia) us-east-1	<a href="#">Bucket and objects not public</a>	December 1, 2023, 16:50:21 (UTC-05:00)
<input type="radio"/>	<a href="#">unnati-first-bucket</a>	US East (N. Virginia) us-east-1	<a href="#">Objects can be public</a>	May 11, 2023, 17:43:24 (UTC-04:00)

- Loaded CSV data into AWS Glue.
- Created 10 AWS Glue Crawlers, one for each table.
- Configured Crawlers to extract metadata from respective table's CSV data.
- Established a database named 'covid\_database' in AWS Glue.
- Associated each Crawler with 'covid\_database' for storing metadata.
- Enabled automation for dynamic data discovery and cataloging.

<input type="checkbox"/>	Name ▾	State ▾	Schedule	Last run ▾	Last run time... ▾	Log	Table change...
<input type="checkbox"/>	<a href="#">enigma_jhud</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">factCovid</a>	✔ Ready		✔ Succeeded	December 2, 202...	<a href="#">View log</a>	-
<input type="checkbox"/>	<a href="#">nytimes-data-in-...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">nytimes-data-in-...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">rearc-covid-19-te...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">rearc-covid-19-te...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">rearc-covid-19-te...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">rearc-usa-hospita...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 updated
<input type="checkbox"/>	<a href="#">static-datasets-co...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">static-datasets-co...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">static-datasets-st...</a>	✔ Ready		✔ Succeeded	December 1, 202...	<a href="#">View log</a>	1 created

## Data Transformation:

Data was transformed following the warehouse design.



Utilizing the boto3 library, systematically retrieves tables from an S3 bucket. Subsequently, these tables undergo a structured ETL process, and the refined results are methodically stored in the designated dwbireturn S3 bucket. This streamlined workflow embodies an efficient and automated data management and transformation process within the AWS environment.

```
[1]: import boto3
import pandas as pd
from io import StringIO
```

```
[4]: import time
import pandas as pd
import boto3
from botocore.exceptions import ClientError

def download_and_load_query_results(
    client: boto3.client, query_response: dict
) -> pd.DataFrame:
    query_execution_id = query_response["QueryExecutionId"]
    while True:
        try:
            # Check query execution status
            query_status = client.get_query_execution(
                QueryExecutionId=query_execution_id
            )["QueryExecution"]["Status"]["State"]
            if query_status == 'SUCCEEDED':
                break
            elif query_status in ['FAILED', 'CANCELLED']:
                raise ValueError(f"Query execution {query_status}")

            time.sleep(1) # Wait for 1 second before checking again
        except ClientError as e:
            raise e

    temp_file_location = "athena_query_results.csv"
    s3_client = boto3.client(
```

This process involves fetching files from Amazon S3, querying the data in those files using Amazon Athena's SQL capabilities, and then using the query results for data modeling. This approach leverages the serverless architecture of S3 and Athena, Efficiently managing large datasets in the cloud demands scalability and cost-effectiveness. Leveraging cloud solutions allows organizations to seamlessly scale resources, optimize data querying, and implement effective data modeling. This approach ensures streamlined operations and cost savings by adapting to evolving business needs

The data was read using the queries:

```
[7]: response = athena_client.start_query_execution(
    QueryString="SELECT * FROM countrycode",
    QueryExecutionContext={"Database": SCHEMA_NAME},
    ResultConfiguration={
        "OutputLocation": S3_STAGING_DIR,
        "EncryptionConfiguration": {"EncryptionOption": "SSE_S3"},
    },
)

df_data = download_and_load_query_results(athena_client, response)

[8]: countrycode = download_and_load_query_results(athena_client,response)
```

The table was created as:

**factCovid:**

```
factCovid_1 = enigma_jhud[['fips', 'province_state', 'country_region', 'confirmed', 'deaths', 'recovered', 'active']]
factCovid_2 = states_daily[['fips', 'date', 'positive', 'negative', 'hospitalizedcurrently', 'hospitalized', 'hospitalizeddischarged']]
factCovid = pd.merge(factCovid_1, factCovid_2, on='fips', how='inner')
```

**dimRegion:**

```
dimRegion_1 = enigma_jhud [['fips', 'province_state', 'country_region', 'latitude', 'longitude']]
dimRegion_2 = us_county [['fips', 'county', 'state']]
dimRegion = pd.merge(dimRegion_1, dimRegion_2, on='fips', how='inner')
```

**dimDate:**

```
dimDate = states_daily[['fips', 'date']]
```

```
dimDate['date'] = pd.to_datetime(dimDate['date'], format='%Y%m%d')
```

```
dimDate['year'] = dimDate['date'].dt.year
dimDate['month'] = dimDate['date'].dt.month
dimDate['day_of_week'] = dimDate['date'].dt.dayofweek
```

**dimHospital:**

```
dimHospital = rearc_usa_hospital_beds[['fips', 'state_name', 'latitude', 'longitude', 'hq_address', 'hospital_name', 'hospital_type',
                                       'hq_city', 'hq_state']]
```

The data is now put in the S3 bucket of dwbi-return

## factCovid

```
csv_buffer = StringIO()
factCovid.to_csv(csv_buffer, index=False)

# Specify the S3 bucket and file name
bucket_name = 'dwbi-return'
file_name = 'output-datamodel/factCovid/factCovid.csv'

# Upload the CSV data to S3
s3.put_object(Bucket=bucket_name, Key=file_name, Body=csv_buffer.getvalue())

print(f"File '{file_name}' uploaded to S3 bucket '{bucket_name}'")
```

## dimRegion:

```
csv_buffer = StringIO()
dimRegion.to_csv(csv_buffer, index=False)

# Specify the S3 bucket and file name
bucket_name = 'dwbi-return'
file_name = 'output-datamodel/dimRegion.csv'

# Upload the CSV data to S3
s3.put_object(Bucket=bucket_name, Key=file_name, Body=csv_buffer.getvalue())

print(f"File '{file_name}' uploaded to S3 bucket '{bucket_name}'")
```

## dimDate:

```
csv_buffer = StringIO()
dimDate.to_csv(csv_buffer, index=False)

# Specify the S3 bucket and file name
bucket_name = 'dwbi-return'
file_name = 'output-datamodel/dimDate.csv'

# Upload the CSV data to S3
s3.put_object(Bucket=bucket_name, Key=file_name, Body=csv_buffer.getvalue())

print(f"File '{file_name}' uploaded to S3 bucket '{bucket_name}'")
```





## dimHospital:

```
csv_buffer = StringIO()
dimHospital.to_csv(csv_buffer, index=False)

# Specify the S3 bucket and file name
bucket_name = 'dwbi-return'
file_name = 'output-datamodel/dimHospital.csv'

# Upload the CSV data to S3
s3.put_object(Bucket=bucket_name, Key=file_name, Body=csv_buffer.getvalue())

print(f"File '{file_name}' uploaded to S3 bucket '{bucket_name}'")
```

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 <a href="#">dimDate.csv</a>	csv	December 1, 2023, 20:38:56 (UTC-05:00)	469.3 KB	Standard
<input type="checkbox"/>	 <a href="#">dimHospital.csv</a>	csv	December 1, 2023, 20:38:56 (UTC-05:00)	957.9 KB	Standard
<input type="checkbox"/>	 <a href="#">dimRegion.csv</a>	csv	December 1, 2023, 20:38:27 (UTC-05:00)	2.0 GB	Standard
<input type="checkbox"/>	 <a href="#">factCovid.csv</a>	csv	December 1, 2023, 20:43:54 (UTC-05:00)	1.8 MB	Standard

This transformed data was then pushed to Crawler for the analysis using Athena:  
This bucket was then loaded to the crawler, to perform data analysis using Athena.

## Data Analysis:

Find the cities with hospitals:

# ▼	hq_city ▼	_col1 ▼
1	Phoenix	24
2	West Haven	1
3	Gainesville	8
4	Roseburg	2
5	Prattville	1

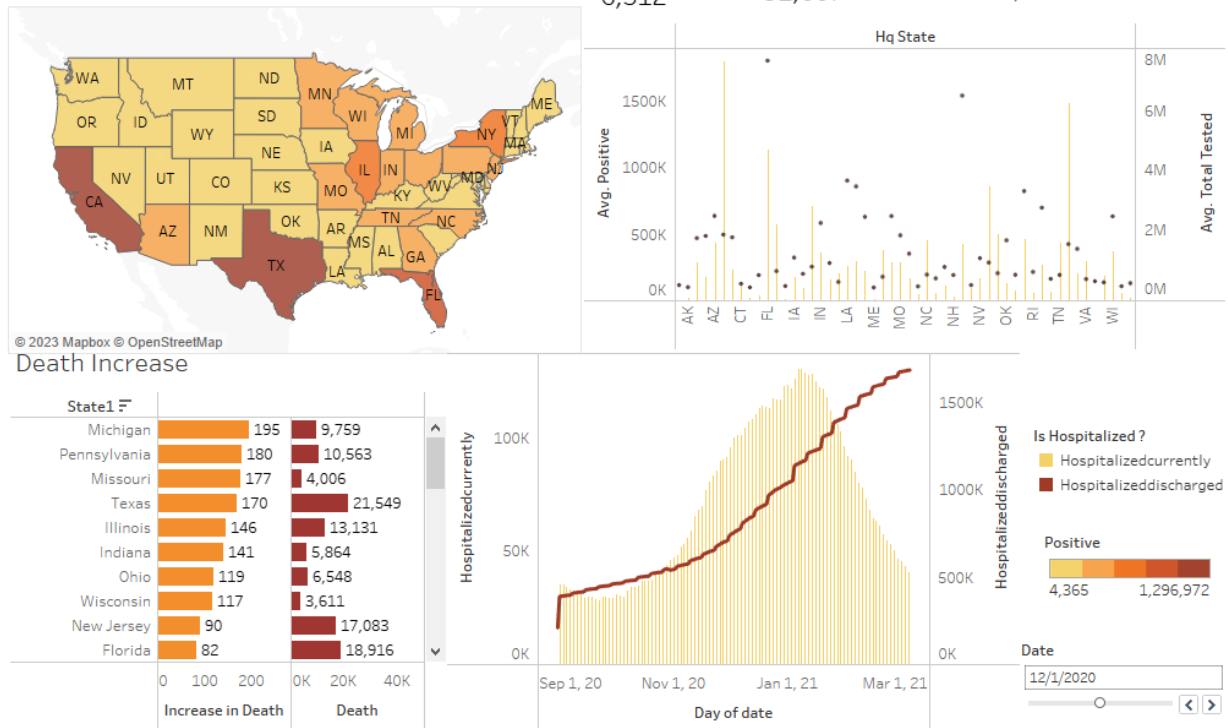
Top State with Highest Deaths:

# ▼	province_state ▼	total_deaths ▼
1	Puerto Rico	1758939.0



## Data Visualization Dashboard:

# Covid Analysis



After completing the AWS part, we connected tableau with AWS to create dashboard. The following KeyPoint that we covered in the visualization:

1. Developed an interactive US country map to examine the distribution of COVID-19 cases across various states. Incorporated state and date filters into the dashboard to facilitate tracking of daily case trends in different regions.
2. Implemented calculated fields to determine the positivity rate in each state through a second graph. Utilized the ratio of positive cases to total cases as a parameter. The dashboard allows users to apply date and state filters for refined data analysis.
3. Constructed a horizontal bar graph to visualize the daily increase in deaths versus total deaths in each state.
4. Designed a bar-line chart to depict the daily hospitalization of patients and the daily discharge of patients over a two-year period. Notably, the chart illustrates a gradual increase in discharged patients over time.

**5. Identified persistent leaders in COVID-19 impact, such as New York, California, Michigan, Pennsylvania, Missouri, Texas, Illinois, New Jersey, Florida, and Ohio. These states consistently ranked high and faced financial repercussions due to the pandemic.**

**6. Despite the United States boasting a top-tier healthcare infrastructure, we discovered shortages of hospital beds and ICUs during the peak of the COVID-19 pandemic, notably in states such as Texas and California.**