Project Report

Summer-2020

Topic:

Analysis of Covid-19 Datasets using AWS SageMaker

Ruchi Ninawe

U72686676

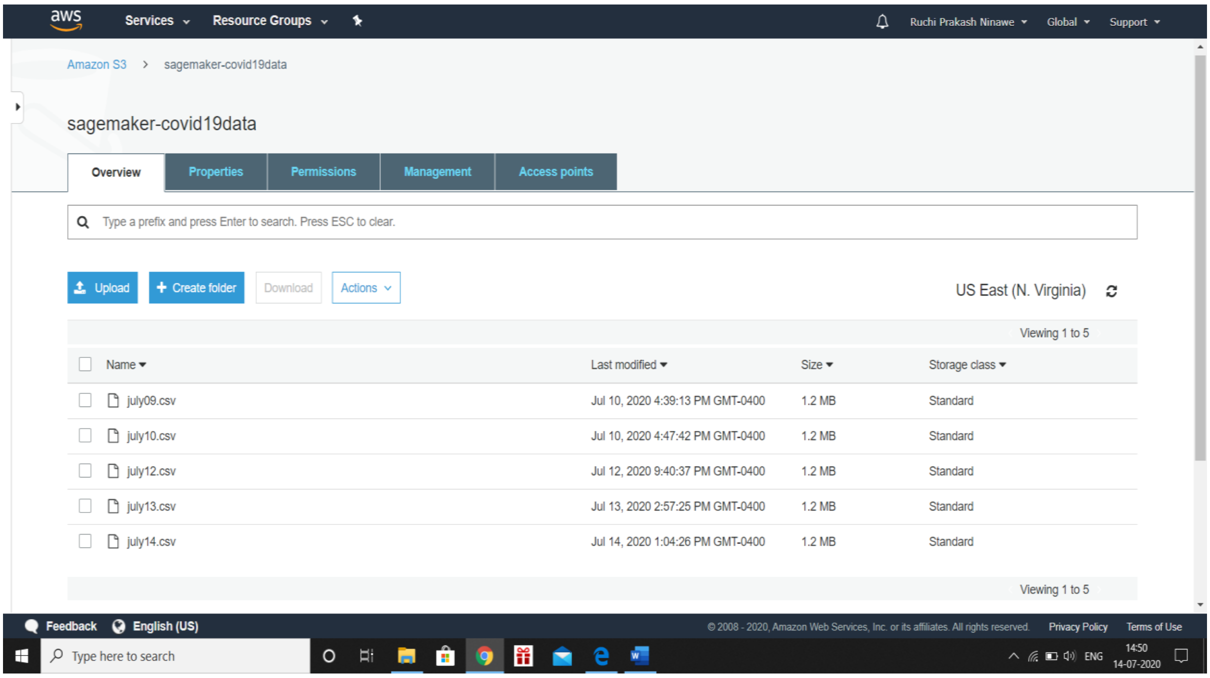
Guided by Dr Phil Ventura, Professor, University of South Florida

Project Description

The purpose of this project is to study, manage and analyse a few COVID-19 datasets using AWS SageMaker. This project includes the following elements:

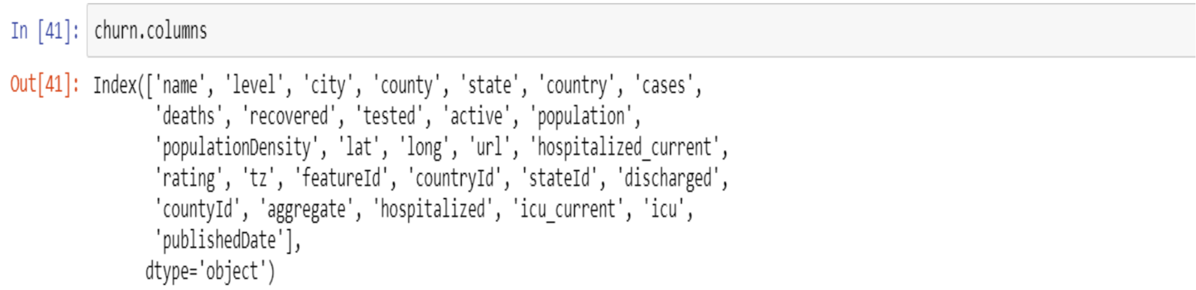
- There are five source files(.csv) in the S3(*sagemaker-covid19data*) bucket used for this project.
- A python file(data_process) written in *jupyter*

S3 Bucket



Name	Last modified	Size	Storage class
july09.csv	Jul 10, 2020 4:39:13 PM GMT-0400	1.2 MB	Standard
july10.csv	Jul 10, 2020 4:47:42 PM GMT-0400	1.2 MB	Standard
july12.csv	Jul 12, 2020 9:40:37 PM GMT-0400	1.2 MB	Standard
july13.csv	Jul 13, 2020 2:57:25 PM GMT-0400	1.2 MB	Standard
july14.csv	Jul 14, 2020 1:04:26 PM GMT-0400	1.2 MB	Standard

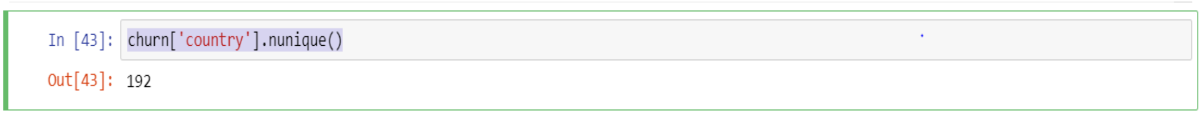
These files contain data files[1] from 192 countries across the world, each file containing the name of the country, cases, deaths, tested, recovered, etc. These datasets are trusted and provided by the *covid-19 datalake*[2] provided by AWS.



```
In [41]: churn.columns

Out[41]: Index(['name', 'level', 'city', 'county', 'state', 'country', 'cases',
                'deaths', 'recovered', 'tested', 'active', 'population',
                'populationDensity', 'lat', 'long', 'url', 'hospitalized_current',
                'rating', 'tz', 'featureId', 'countryId', 'stateId', 'discharged',
                'countyId', 'aggregate', 'hospitalized', 'icu_current', 'icu',
                'publishedDate'],
               dtype='object')
```

In order to find the number of unique countries, I have used the “*churn['country'].nunique()*” command.

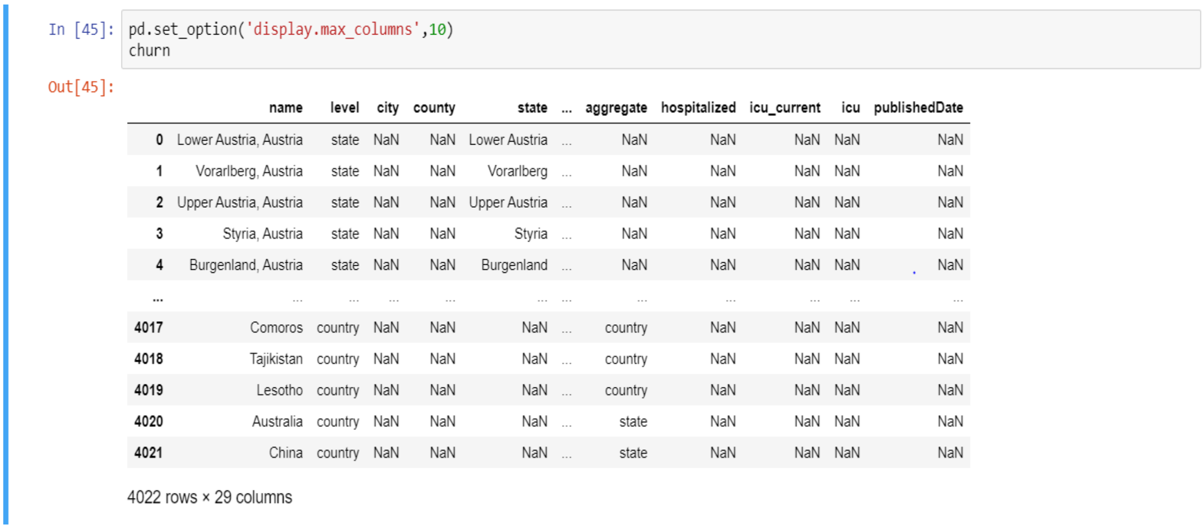


```
In [43]: churn['country'].nunique()
Out[43]: 192
```

Python file in *jupyter_notebook*: *data_process.ipynb*

This python file manages and analyses the data sets from S3 bucket. The code for the analysis of the dataset from July 09, 2020 is provided namely *data_process09.pdf* for reference.

The dataset imported from S3 is displayed:



```
In [45]: pd.set_option('display.max_columns',10)
         churn
Out[45]:
```

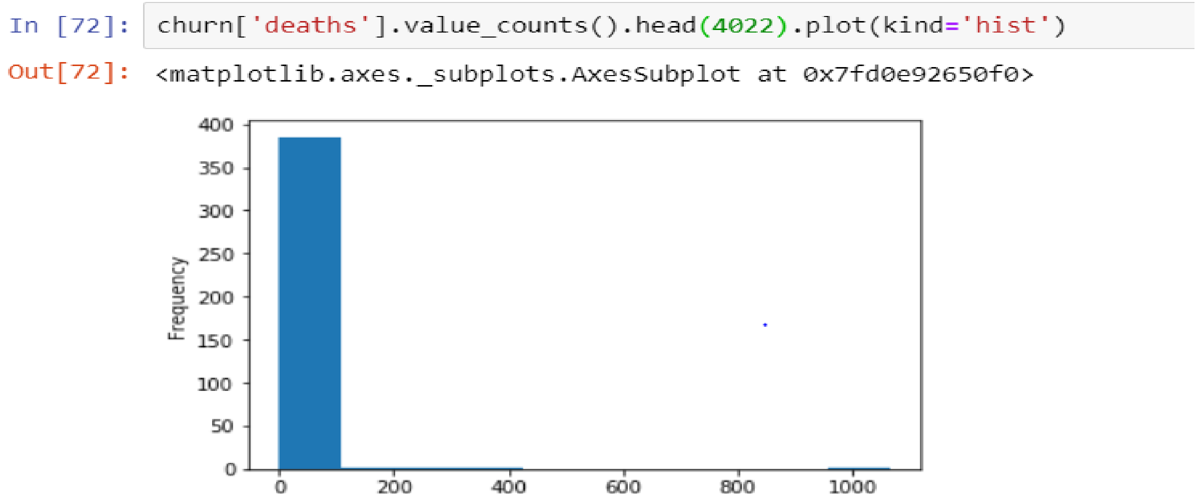
	name	level	city	county	state	...	aggregate	hospitalized	icu_current	icu	publishedDate
0	Lower Austria, Austria	state	NaN	NaN	Lower Austria	...	NaN	NaN	NaN	NaN	NaN
1	Vorarlberg, Austria	state	NaN	NaN	Vorarlberg	...	NaN	NaN	NaN	NaN	NaN
2	Upper Austria, Austria	state	NaN	NaN	Upper Austria	...	NaN	NaN	NaN	NaN	NaN
3	Styria, Austria	state	NaN	NaN	Styria	...	NaN	NaN	NaN	NaN	NaN
4	Burgenland, Austria	state	NaN	NaN	Burgenland	...	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
4017	Comoros	country	NaN	NaN	NaN	...	country	NaN	NaN	NaN	NaN
4018	Tajikistan	country	NaN	NaN	NaN	...	country	NaN	NaN	NaN	NaN
4019	Lesotho	country	NaN	NaN	NaN	...	country	NaN	NaN	NaN	NaN
4020	Australia	country	NaN	NaN	NaN	...	state	NaN	NaN	NaN	NaN
4021	China	country	NaN	NaN	NaN	...	state	NaN	NaN	NaN	NaN

4022 rows × 29 columns

Analysis

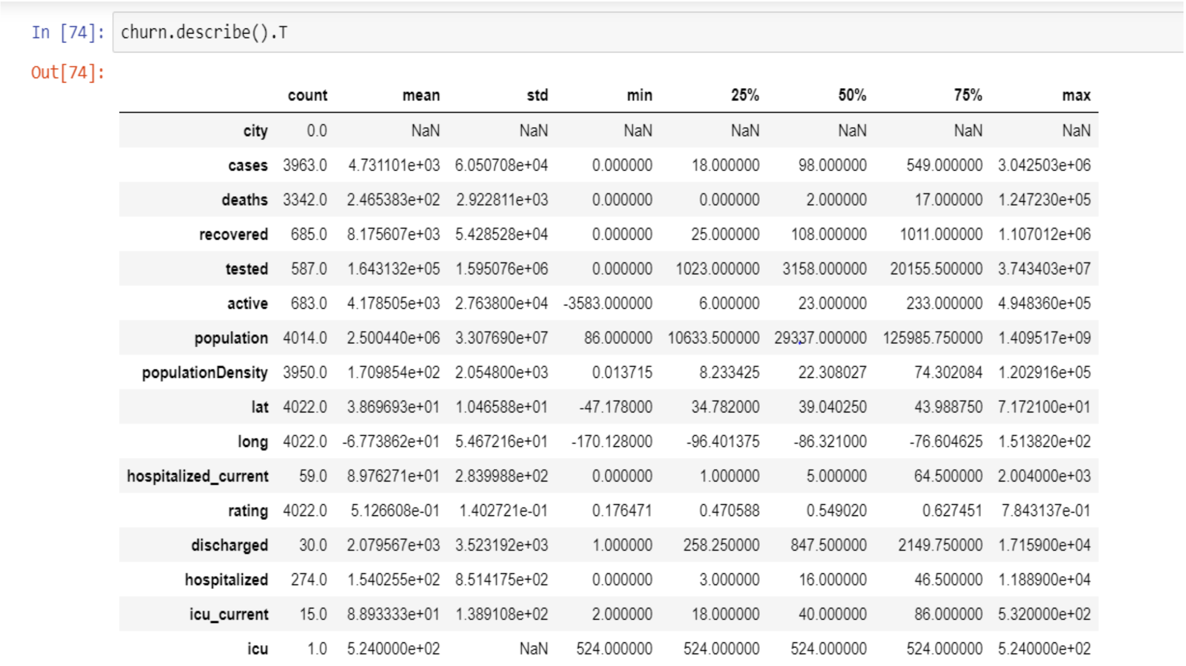
1. **July 09,2020**

The graph below shows a histogram for a total of 4022 entries for the frequency of deaths on July 09.



```
In [72]: churn['deaths'].value_counts().head(4022).plot(kind='hist')
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd0e92650f0>
```

The numerical analysis is shown below:

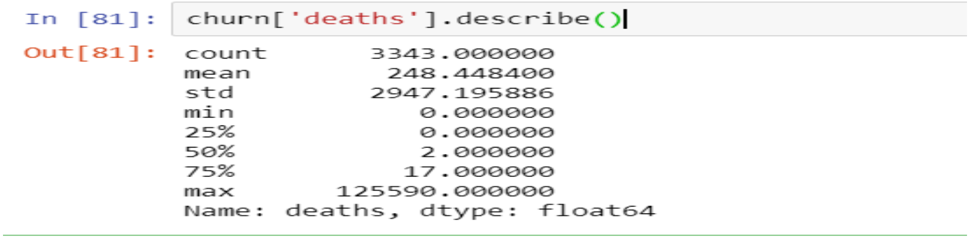


```
In [74]: churn.describe().T
Out[74]:
```

	count	mean	std	min	25%	50%	75%	max
city	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cases	3963.0	4.731101e+03	6.050708e+04	0.000000	18.000000	98.000000	549.000000	3.042503e+06
deaths	3342.0	2.465383e+02	2.922811e+03	0.000000	0.000000	2.000000	17.000000	1.247230e+05
recovered	685.0	8.175607e+03	5.428528e+04	0.000000	25.000000	108.000000	1011.000000	1.107012e+06
tested	587.0	1.643132e+05	1.595076e+06	0.000000	1023.000000	3158.000000	20155.500000	3.743403e+07
active	683.0	4.178505e+03	2.763800e+04	-3583.000000	6.000000	23.000000	233.000000	4.948360e+05
population	4014.0	2.500440e+06	3.307690e+07	86.000000	10633.500000	29337.000000	125985.750000	1.409517e+09
populationDensity	3950.0	1.709854e+02	2.054800e+03	0.013715	8.233425	22.308027	74.302084	1.202916e+05
lat	4022.0	3.869693e+01	1.046588e+01	-47.178000	34.782000	39.040250	43.988750	7.172100e+01
long	4022.0	-6.773862e+01	5.467216e+01	-170.128000	-96.401375	-86.321000	-76.604625	1.513820e+02
hospitalized_current	59.0	8.976271e+01	2.839988e+02	0.000000	1.000000	5.000000	64.500000	2.004000e+03
rating	4022.0	5.126608e-01	1.402721e-01	0.176471	0.470588	0.549020	0.627451	7.843137e-01
discharged	30.0	2.079567e+03	3.523192e+03	1.000000	258.250000	847.500000	2149.750000	1.715900e+04
hospitalized	274.0	1.540255e+02	8.514175e+02	0.000000	3.000000	16.000000	46.500000	1.188900e+04
icu_current	15.0	8.893333e+01	1.389108e+02	2.000000	18.000000	40.000000	86.000000	5.320000e+02
icu	1.0	5.240000e+02	NaN	524.000000	524.000000	524.000000	524.000000	5.240000e+02

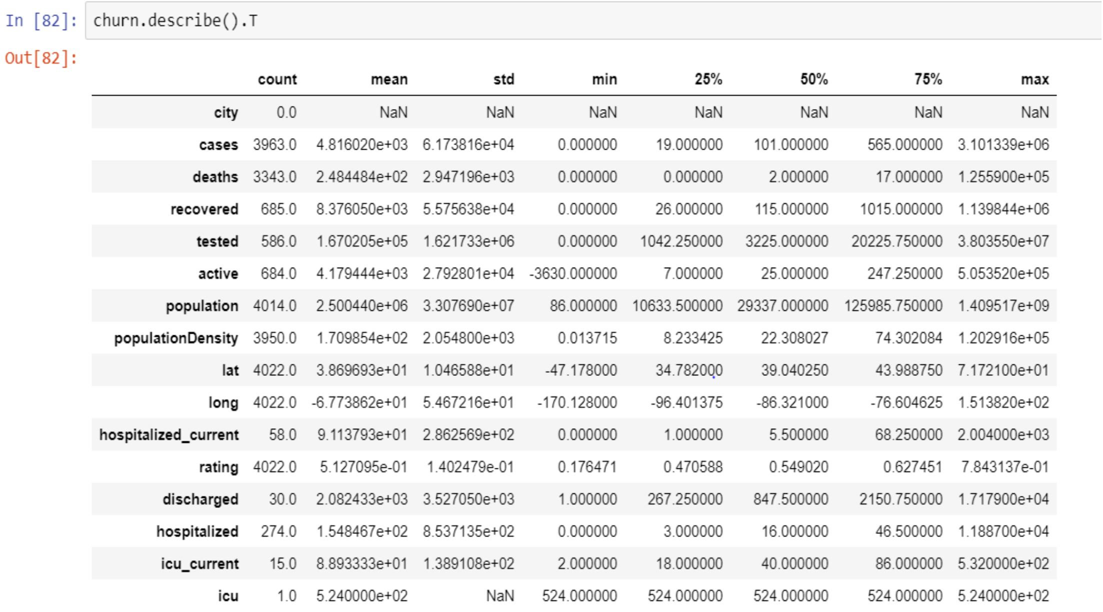
2. **July 10, 2020**

   The numerical analysis for the rate of deaths is shown below:



```
In [81]: churn['deaths'].describe()
Out[81]: count         3343.000000
         mean           248.448400
         std           2947.195886
         min              0.000000
         25%              0.000000
         50%              2.000000
         75%             17.000000
         max         125590.000000
         Name: deaths, dtype: float64
```

The numerical analysis for all the columns is shown below:

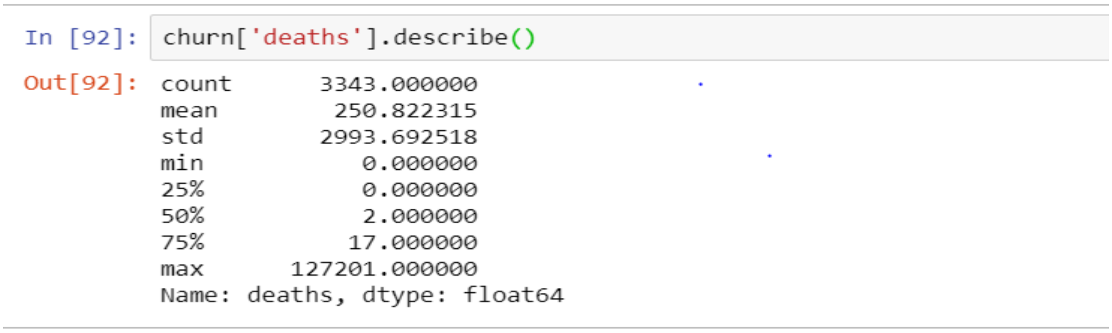


```
In [82]: churn.describe().T
Out[82]:
```

	count	mean	std	min	25%	50%	75%	max
city	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cases	3963.0	4.816020e+03	6.173816e+04	0.000000	19.000000	101.000000	565.000000	3.101339e+06
deaths	3343.0	2.484484e+02	2.947196e+03	0.000000	0.000000	2.000000	17.000000	1.255900e+05
recovered	685.0	8.376050e+03	5.575638e+04	0.000000	26.000000	115.000000	1015.000000	1.139844e+06
tested	586.0	1.670205e+05	1.621733e+06	0.000000	1042.250000	3225.000000	20225.750000	3.803550e+07
active	684.0	4.179444e+03	2.792801e+04	-3630.000000	7.000000	25.000000	247.250000	5.053520e+05
population	4014.0	2.500440e+06	3.307690e+07	86.000000	10633.500000	29337.000000	125985.750000	1.409517e+09
populationDensity	3950.0	1.709854e+02	2.054800e+03	0.013715	8.233425	22.308027	74.302084	1.202916e+05
lat	4022.0	3.869693e+01	1.046588e+01	-47.178000	34.782000	39.040250	43.988750	7.172100e+01
long	4022.0	-6.773862e+01	5.467216e+01	-170.128000	-96.401375	-86.321000	-76.604625	1.513820e+02
hospitalized_current	58.0	9.113793e+01	2.862569e+02	0.000000	1.000000	5.500000	68.250000	2.004000e+03
rating	4022.0	5.127095e-01	1.402479e-01	0.176471	0.470588	0.549020	0.627451	7.843137e-01
discharged	30.0	2.082433e+03	3.527050e+03	1.000000	267.250000	847.500000	2150.750000	1.717900e+04
hospitalized	274.0	1.548467e+02	8.537135e+02	0.000000	3.000000	16.000000	46.500000	1.188700e+04
icu_current	15.0	8.893333e+01	1.389108e+02	2.000000	18.000000	40.000000	86.000000	5.320000e+02
icu	1.0	5.240000e+02	NaN	524.000000	524.000000	524.000000	524.000000	5.240000e+02

3. **July 12, 2020**

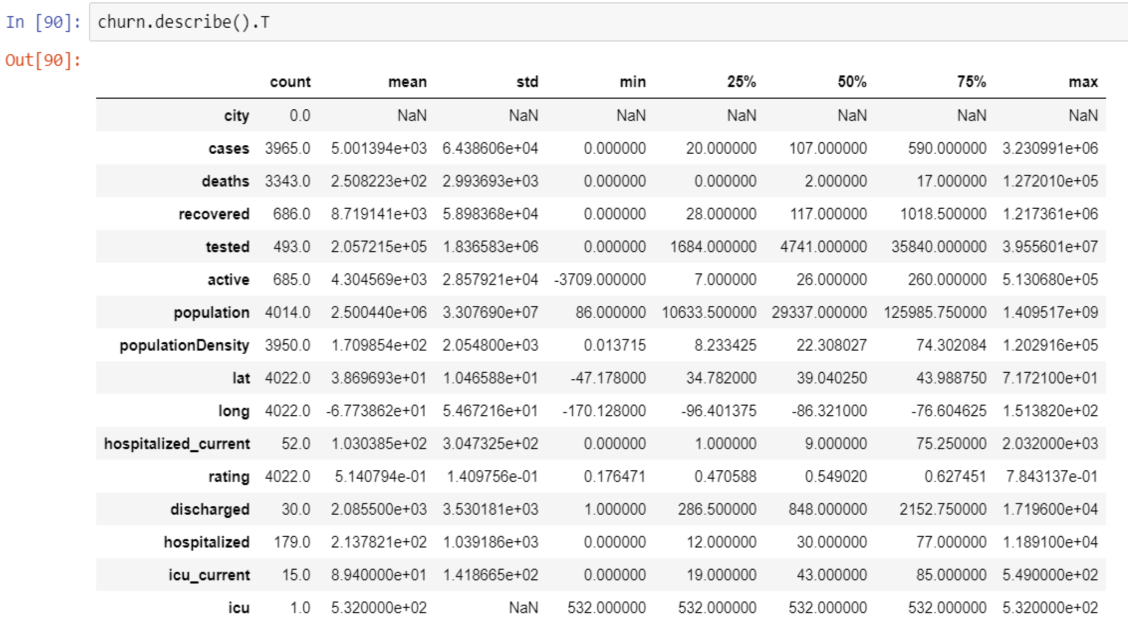
The numerical analysis for the rate of deaths is shown below:



```
In [92]: churn['deaths'].describe()

Out[92]: count       3343.000000
         mean         250.822315
         std         2993.692518
         min            0.000000
         25%            0.000000
         50%            2.000000
         75%           17.000000
         max       127201.000000
         Name: deaths, dtype: float64
```

The numerical analysis for all the columns is shown below:



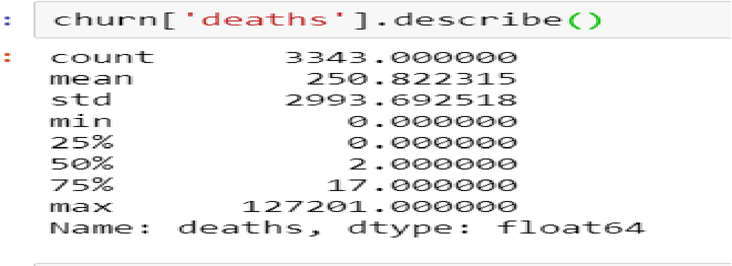
```
In [90]: churn.describe().T

Out[90]:
```

	count	mean	std	min	25%	50%	75%	max
city	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cases	3965.0	5.001394e+03	6.438606e+04	0.000000	20.000000	107.000000	590.000000	3.230991e+06
deaths	3343.0	2.508223e+02	2.993693e+03	0.000000	0.000000	2.000000	17.000000	1.272010e+05
recovered	686.0	8.719141e+03	5.898368e+04	0.000000	28.000000	117.000000	1018.500000	1.217361e+06
tested	493.0	2.057215e+05	1.836583e+06	0.000000	1684.000000	4741.000000	35840.000000	3.955601e+07
active	685.0	4.304569e+03	2.857921e+04	-3709.000000	7.000000	26.000000	260.000000	5.130680e+05
population	4014.0	2.500440e+06	3.307690e+07	86.000000	10633.500000	29337.000000	125985.750000	1.409517e+09
populationDensity	3950.0	1.709854e+02	2.054800e+03	0.013715	8.233425	22.308027	74.302084	1.202916e+05
lat	4022.0	3.869693e+01	1.046588e+01	-47.178000	34.782000	39.040250	43.988750	7.172100e+01
long	4022.0	-6.773862e+01	5.467216e+01	-170.128000	-96.401375	-86.321000	-76.604625	1.513820e+02
hospitalized_current	52.0	1.030385e+02	3.047325e+02	0.000000	1.000000	9.000000	75.250000	2.032000e+03
rating	4022.0	5.140794e-01	1.409756e-01	0.176471	0.470588	0.549020	0.627451	7.843137e-01
discharged	30.0	2.085500e+03	3.530181e+03	1.000000	286.500000	848.000000	2152.750000	1.719600e+04
hospitalized	179.0	2.137821e+02	1.039186e+03	0.000000	12.000000	30.000000	77.000000	1.189100e+04
icu_current	15.0	8.940000e+01	1.418665e+02	0.000000	19.000000	43.000000	85.000000	5.490000e+02
icu	1.0	5.320000e+02	NaN	532.000000	532.000000	532.000000	532.000000	5.320000e+02

4. **July 13, 2020**

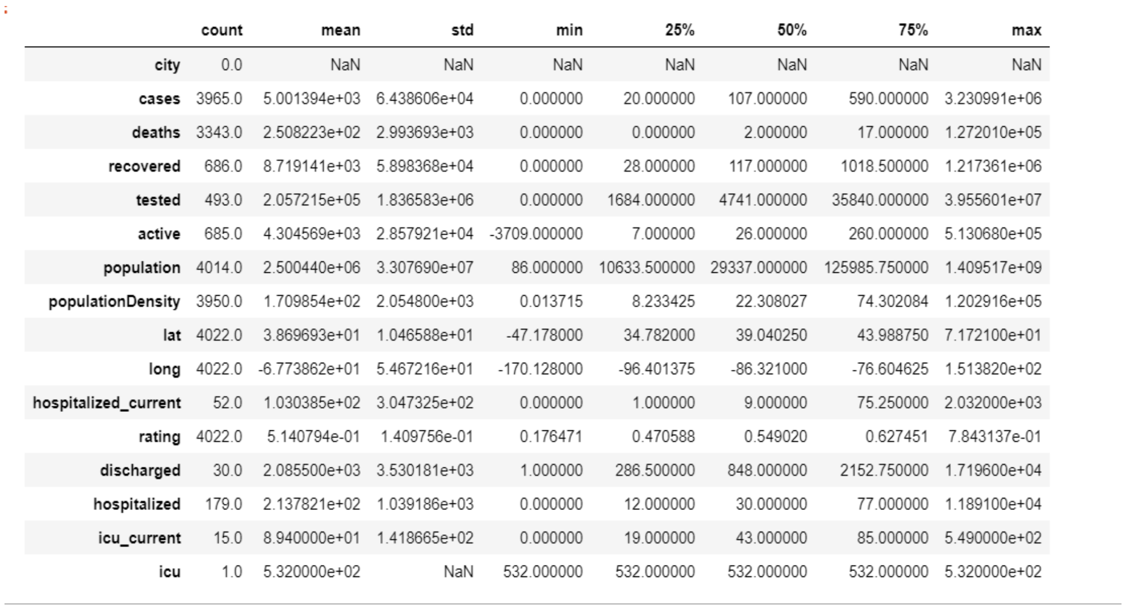
The numerical analysis for the rate of deaths is shown below:



```
: churn['deaths'].describe()

: count       3343.000000
  mean         250.822315
  std         2993.692518
  min            0.000000
  25%            0.000000
  50%            2.000000
  75%           17.000000
  max       127201.000000
  Name: deaths, dtype: float64
```

The numerical analysis for all the columns is shown below:



	count	mean	std	min	25%	50%	75%	max
city	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cases	3965.0	5.001394e+03	6.438606e+04	0.000000	20.000000	107.000000	590.000000	3.230991e+06
deaths	3343.0	2.508223e+02	2.993693e+03	0.000000	0.000000	2.000000	17.000000	1.272010e+05
recovered	686.0	8.719141e+03	5.898368e+04	0.000000	28.000000	117.000000	1018.500000	1.217361e+06
tested	493.0	2.057215e+05	1.836583e+06	0.000000	1684.000000	4741.000000	35840.000000	3.955601e+07
active	685.0	4.304569e+03	2.857921e+04	-3709.000000	7.000000	26.000000	260.000000	5.130680e+05
population	4014.0	2.500440e+06	3.307690e+07	86.000000	10633.500000	29337.000000	125985.750000	1.409517e+09
populationDensity	3950.0	1.709854e+02	2.054800e+03	0.013715	8.233425	22.308027	74.302084	1.202916e+05
lat	4022.0	3.869693e+01	1.046588e+01	-47.178000	34.782000	39.040250	43.988750	7.172100e+01
long	4022.0	-6.773862e+01	5.467216e+01	-170.128000	-96.401375	-86.321000	-76.604625	1.513820e+02
hospitalized_current	52.0	1.030385e+02	3.047325e+02	0.000000	1.000000	9.000000	75.250000	2.032000e+03
rating	4022.0	5.140794e-01	1.409756e-01	0.176471	0.470588	0.549020	0.627451	7.843137e-01
discharged	30.0	2.085500e+03	3.530181e+03	1.000000	286.500000	848.000000	2152.750000	1.719600e+04
hospitalized	179.0	2.137821e+02	1.039186e+03	0.000000	12.000000	30.000000	77.000000	1.189100e+04
icu_current	15.0	8.940000e+01	1.418665e+02	0.000000	19.000000	43.000000	85.000000	5.490000e+02
icu	1.0	5.320000e+02	NaN	532.000000	532.000000	532.000000	532.000000	5.320000e+02

Conclusion

Hence, I created a python file in order to analyse four COVID-19 datasets as it is the most current issues using an AWS service, the SageMaker. For results, I have shown the numerical analysis for all the columns in the dataset to get the mean and median data for all the columns.

References

[1] https://coronadatascraper.com/#data.csv

[2] https://aws.amazon.com/marketplace/search/results?x=0&y=0&searchTerms=covid