Impacts of Covid-19 vs. SARS

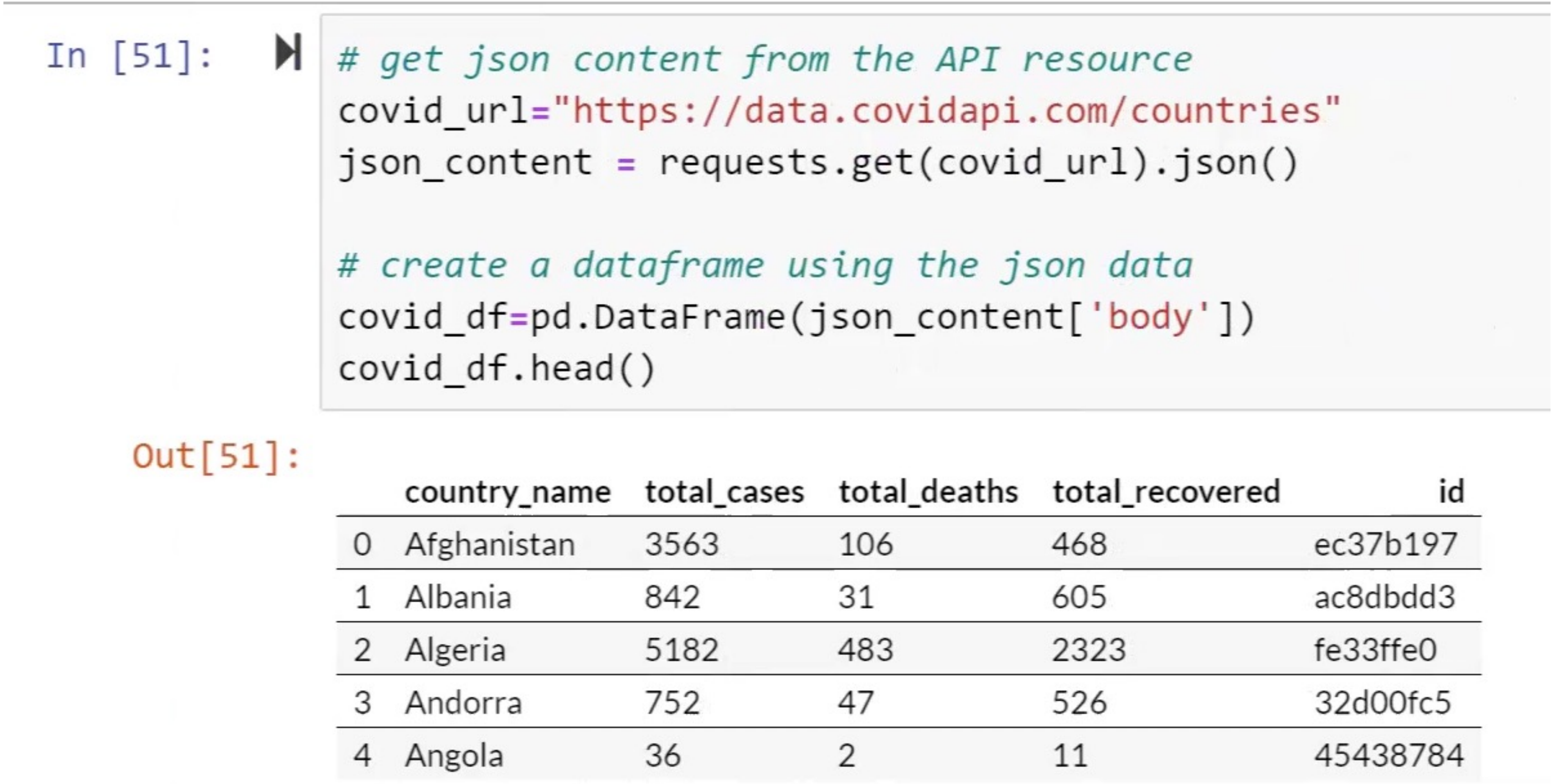
Analysis workflow

1. Acquire Covid-19 and SARS data
    - Sources of data include API, flat file, and a relational database
2. Perform data cleaning and transformation.
    - Check for empty row, remove unnecessary columns, convert wide to long, convert columns to date type.
3. Data Analysis
    - Covid-19 timeline analysis
    - Countries affected by both Covid-19 and SARS
    - Covid-19 and SARS comparative analysis in the first four months.
    - Covid-19 and SARS death rate analysis
4. Data modeling
    - Creating a Linear Regression model using Covid-19 data
    - Use the model to graph the prediction for the next 20 days

Step 1. Import data

Covid-19

- Get Covi-19 total data from API



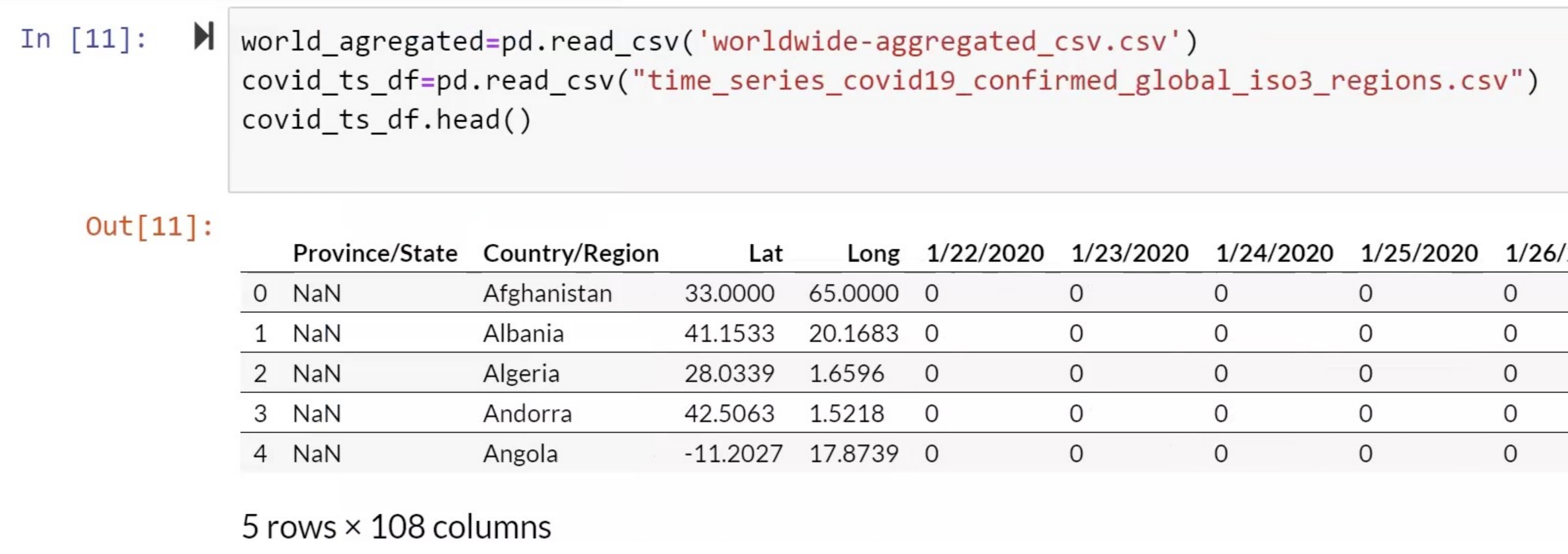
In [51]:

```
# get json content from the API resource
covid_url="https://data.covidapi.com/countries"
json_content = requests.get(covid_url).json()

# create a dataframe using the json data
covid_df=pd.DataFrame(json_content['body'])
covid_df.head()
```

Out[51]:

	country_name	total_cases	total_deaths	total_recovered	id
0	Afghanistan	3563	106	468	ec37b197
1	Albania	842	31	605	ac8dbdd3
2	Algeria	5182	483	2323	fe33ffe0
3	Andorra	752	47	526	32d00fc5
4	Angola	36	2	11	45438784

- Get corona virus timeseries data from flat files



In [11]:

```
world_agregated=pd.read_csv('worldwide-aggregated_csv.csv')
covid_ts_df=pd.read_csv("time_series_covid19_confirmed_global_iso3_regions.csv")
covid_ts_df.head()
```

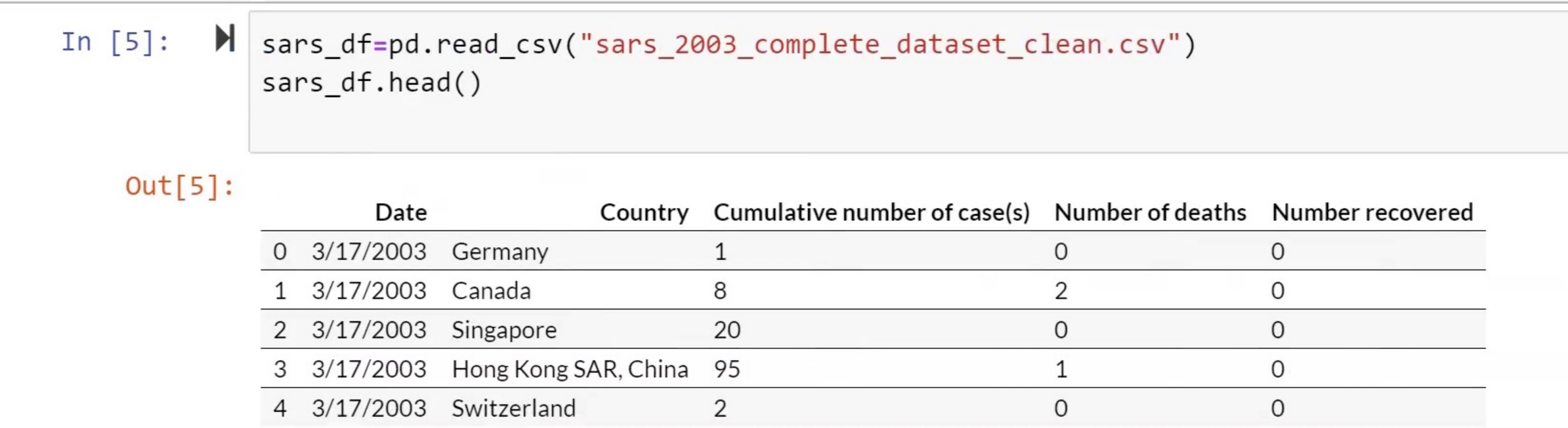
Out[11]:

	Province/State	Country/Region	Lat	Long	1/22/2020	1/23/2020	1/24/2020	1/25/2020	1/26/
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0

5 rows × 108 columns

SARS

- Get SARS total data from a flat file



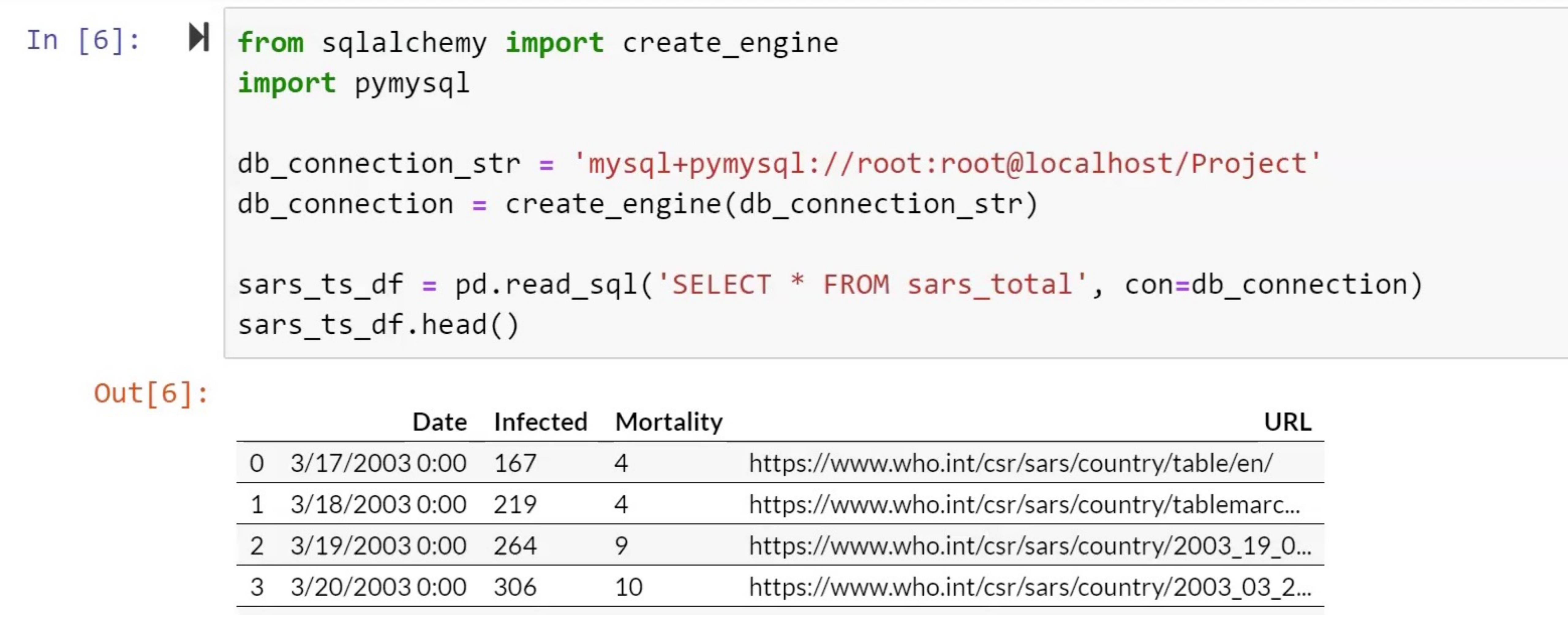
In [5]:

```
sars_df=pd.read_csv("sars_2003_complete_dataset_clean.csv")
sars_df.head()
```

Out[5]:

	Date	Country	Cumulative number of case(s)	Number of deaths	Number recovered
0	3/17/2003	Germany	1	0	0
1	3/17/2003	Canada	8	2	0
2	3/17/2003	Singapore	20	0	0
3	3/17/2003	Hong Kong SAR, China	95	1	0
4	3/17/2003	Switzerland	2	0	0

- Get SARS timeseries data from a relational database



In [6]:

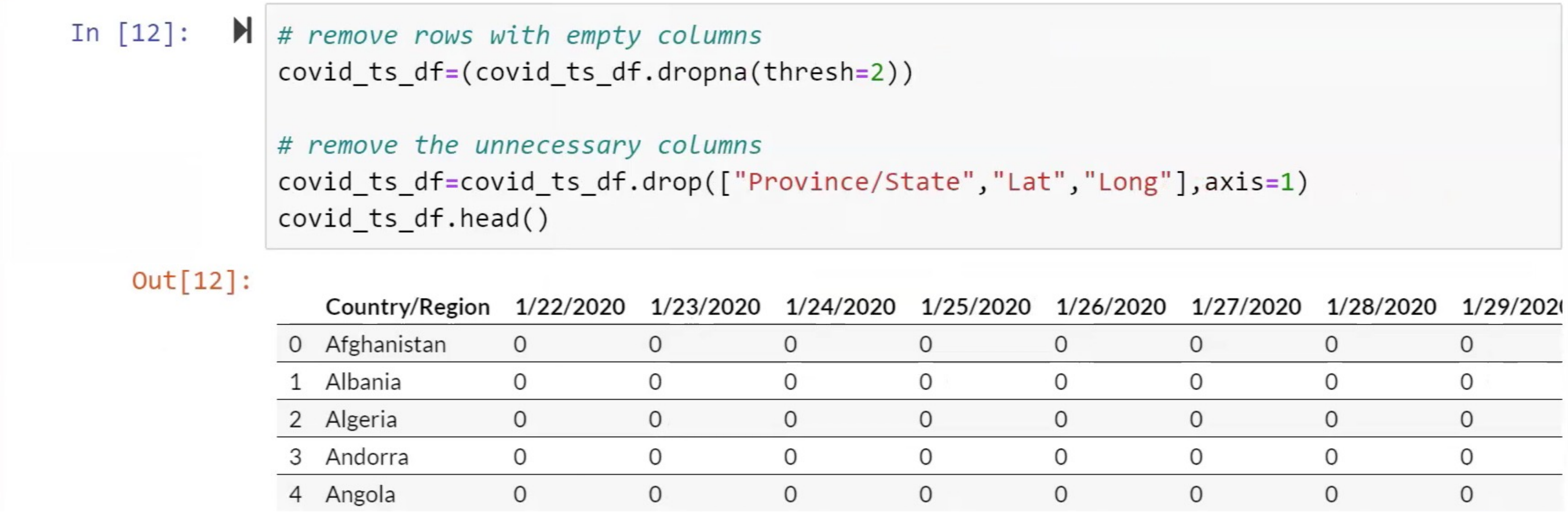
```python
from sqlalchemy import create_engine
import pymysql

db_connection_str = 'mysql+pymysql://root:root@localhost/Project'
db_connection = create_engine(db_connection_str)

sars_ts_df = pd.read_sql('SELECT * FROM sars_total', con=db_connection)
sars_ts_df.head()
```

Out[6]:

	Date	Infected	Mortality	URL
0	3/17/2003 0:00	167	4	https://www.who.int/csr/sars/country/table/en/
1	3/18/2003 0:00	219	4	https://www.who.int/csr/sars/country/tablemarc...
2	3/19/2003 0:00	264	9	https://www.who.int/csr/sars/country/2003_19_0...
3	3/20/2003 0:00	306	10	https://www.who.int/csr/sars/country/2003_03_2...

Step 2. Data Cleaning and Transformation

Remove rows with no values and drop unnecessary columns



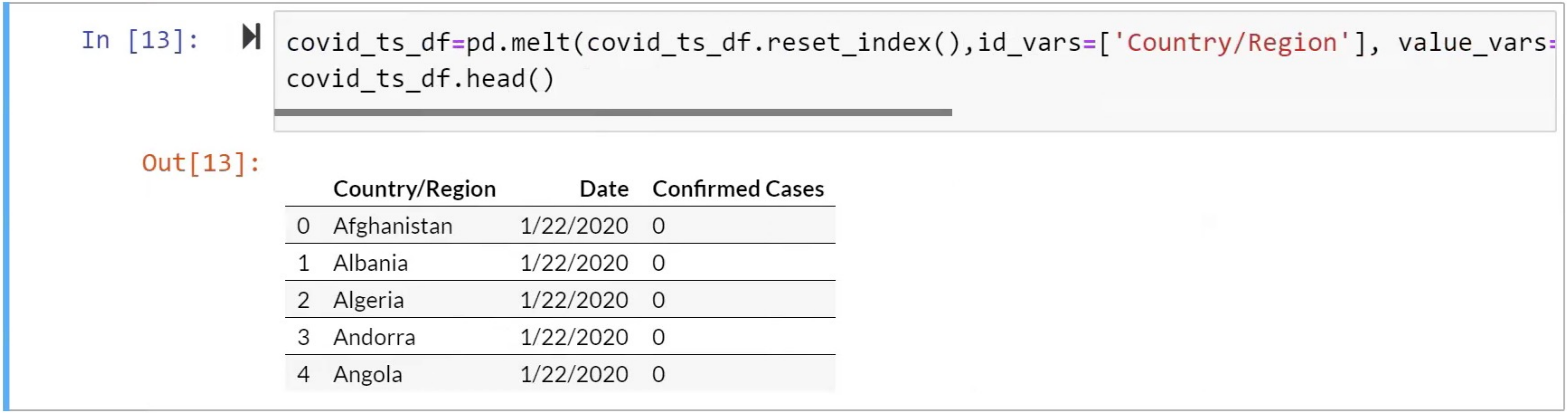
In [12]:

```
# remove rows with empty columns
covid_ts_df=(covid_ts_df.dropna(thresh=2))

# remove the unnecessary columns
covid_ts_df=covid_ts_df.drop(["Province/State","Lat","Long"],axis=1)
covid_ts_df.head()
```

Out[12]:

	Country/Region	1/22/2020	1/23/2020	1/24/2020	1/25/2020	1/26/2020	1/27/2020	1/28/2020	1/29/202
0	Afghanistan	0	0	0	0	0	0	0	0
1	Albania	0	0	0	0	0	0	0	0
2	Algeria	0	0	0	0	0	0	0	0
3	Andorra	0	0	0	0	0	0	0	0
4	Angola	0	0	0	0	0	0	0	0

Convert from wide to long form



In [13]:

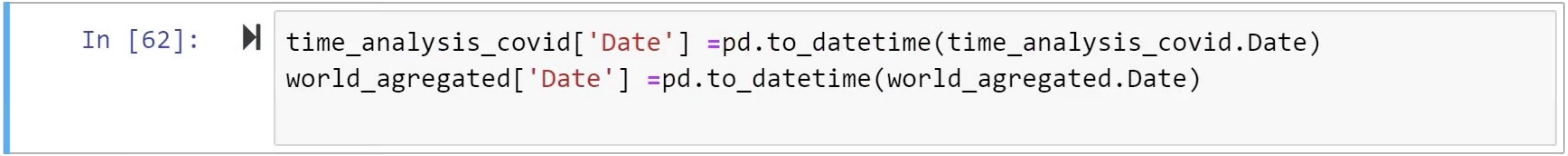
```
covid_ts_df=pd.melt(covid_ts_df.reset_index(),id_vars=['Country/Region'], value_vars=
covid_ts_df.head()
```

Out[13]:

	Country/Region	Date	Confirmed Cases
0	Afghanistan	1/22/2020	0
1	Albania	1/22/2020	0
2	Algeria	1/22/2020	0
3	Andorra	1/22/2020	0
4	Angola	1/22/2020	0

Convert columns to date type

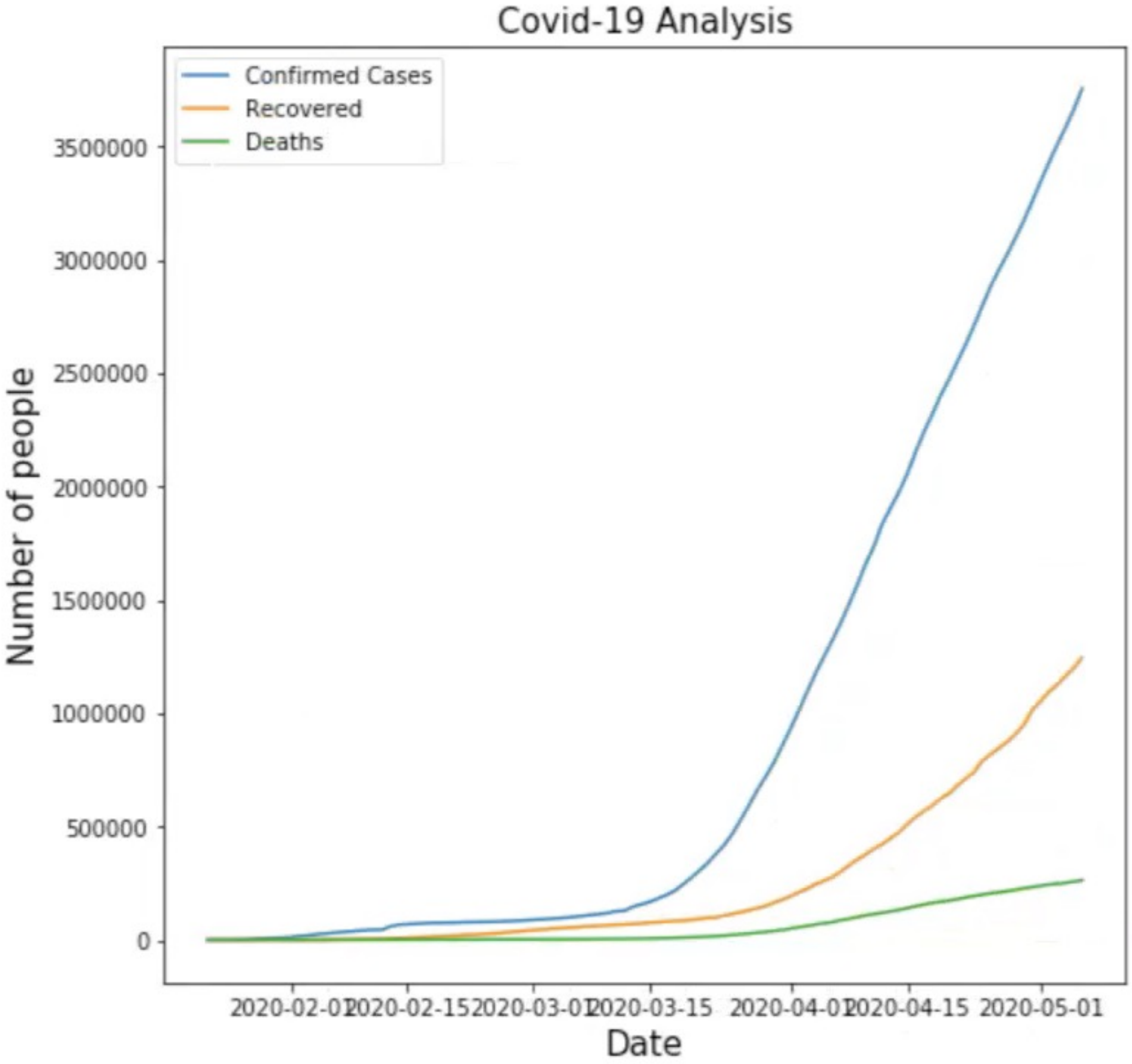
Use the pandas to_datetime function to convert columns that hold date values into Date type



```
time_analysis_covid['Date'] =pd.to_datetime(time_analysis_covid.Date)
world_agregated['Date'] =pd.to_datetime(world_agregated.Date)
```

Step 3. Data Analysis

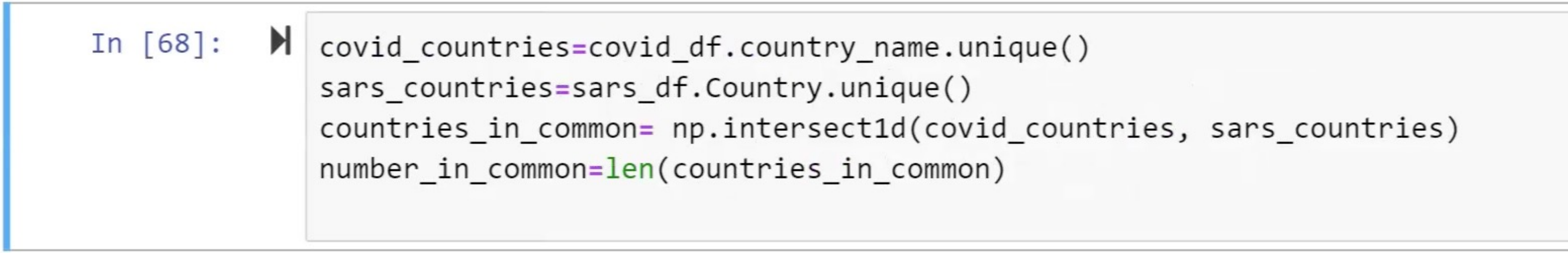
1. Covid-19 Timeline analysis


Covid-19 Analysis
Confirmed Cases
Recovered
Deaths
Number of people
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
2020-02-01
2020-02-15
2020-03-01
2020-03-15
2020-04-01
2020-04-15
2020-05-01
Date

Conclusions

- Even though the number of confirmed cases continues to rise rapidly, the number of recovered cases is also increasing. Furthermore, it is noticeable that the death rate has began stabilizing.

2. Countries affected by both Covid-19 vs. SARS



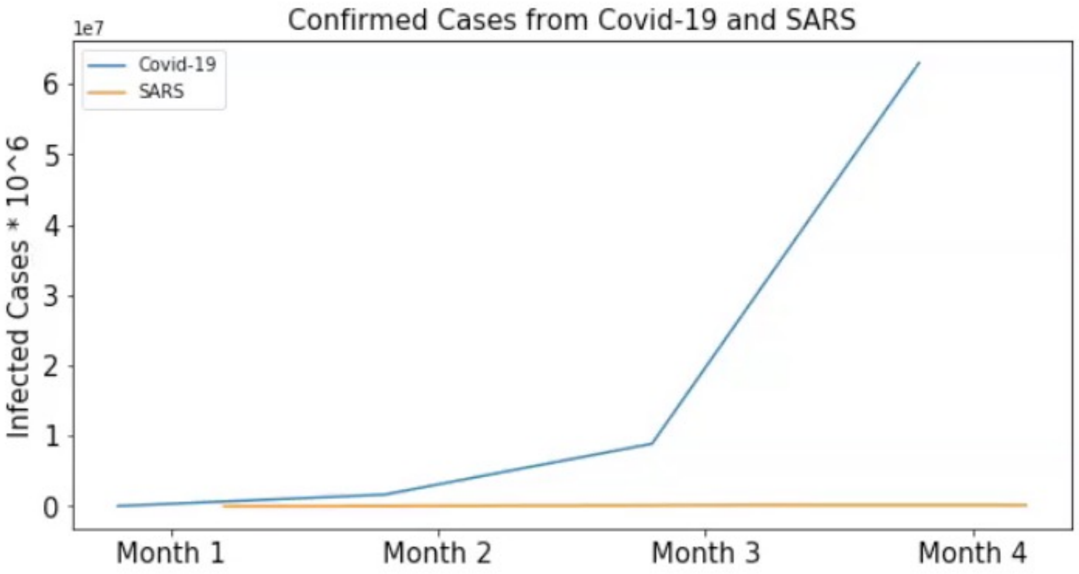
```
covid_countries=covid_df.country_name.unique()
sars_countries=sars_df.Country.unique()
countries_in_common= np.intersect1d(covid_countries, sars_countries)
number_in_common=len(countries_in_common)
```

Conclusion

Countries affected by both Covid-19 and SARS are 29. They include

Australia, Belgium, Brazil, Bulgaria, Canada, China, Colombia, Finland, France, Germany, India, Indonesia, Italy, Japan, Kuwait, Malaysia, Mongolia, New Zealand, Philippines, Poland, Romania, Singapore, Slovenia, South Africa, Spain, Sweden, Switzerland, Thailand, United Kingdom

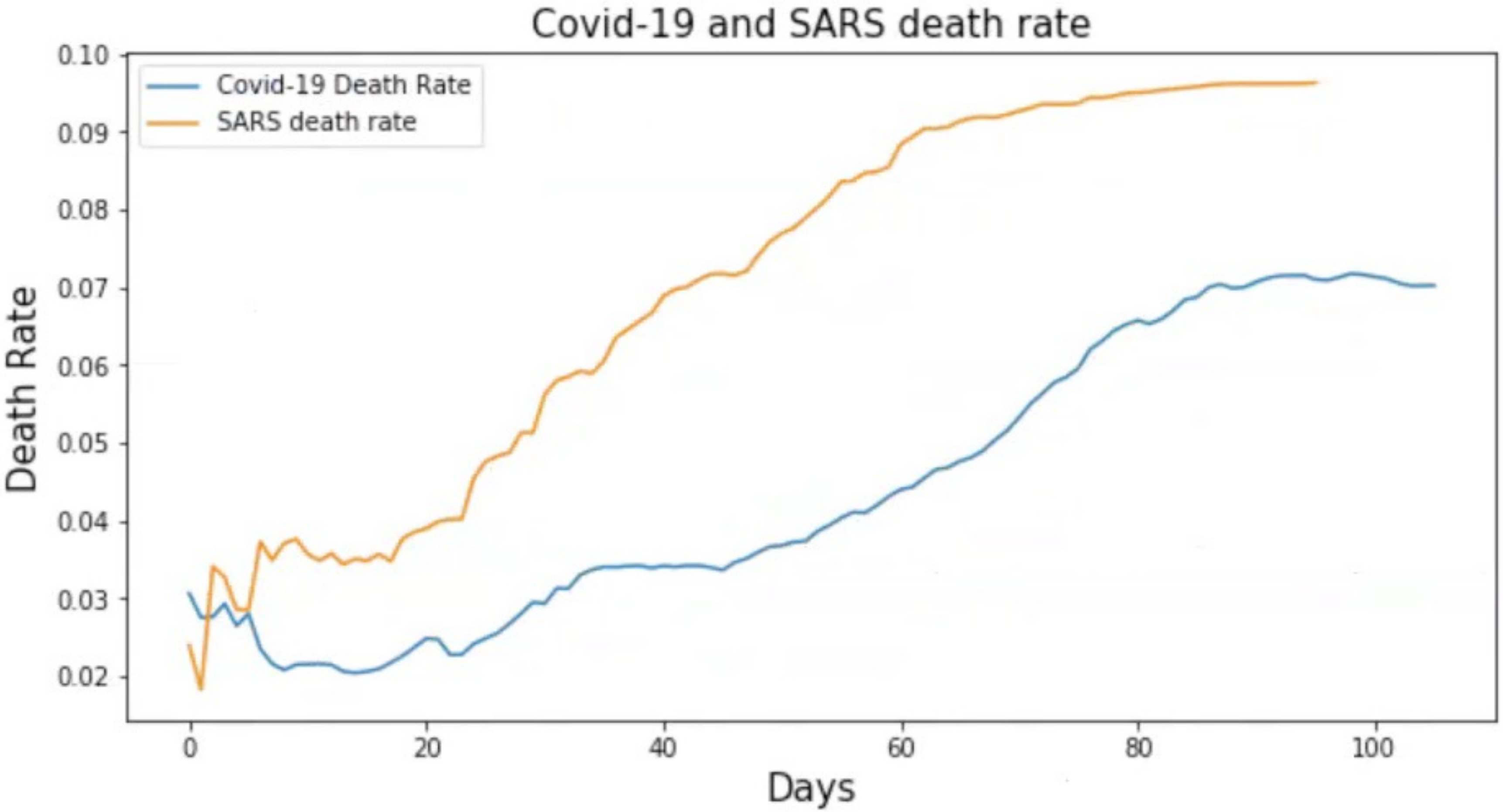
3. Covid-19 vs. SARS comparative analysis in the first four months



Conclusions

- Covid-19 is spreading at a higher and faster rate in comparison to SARS. The SARS confirmed cases are almost insignificant.

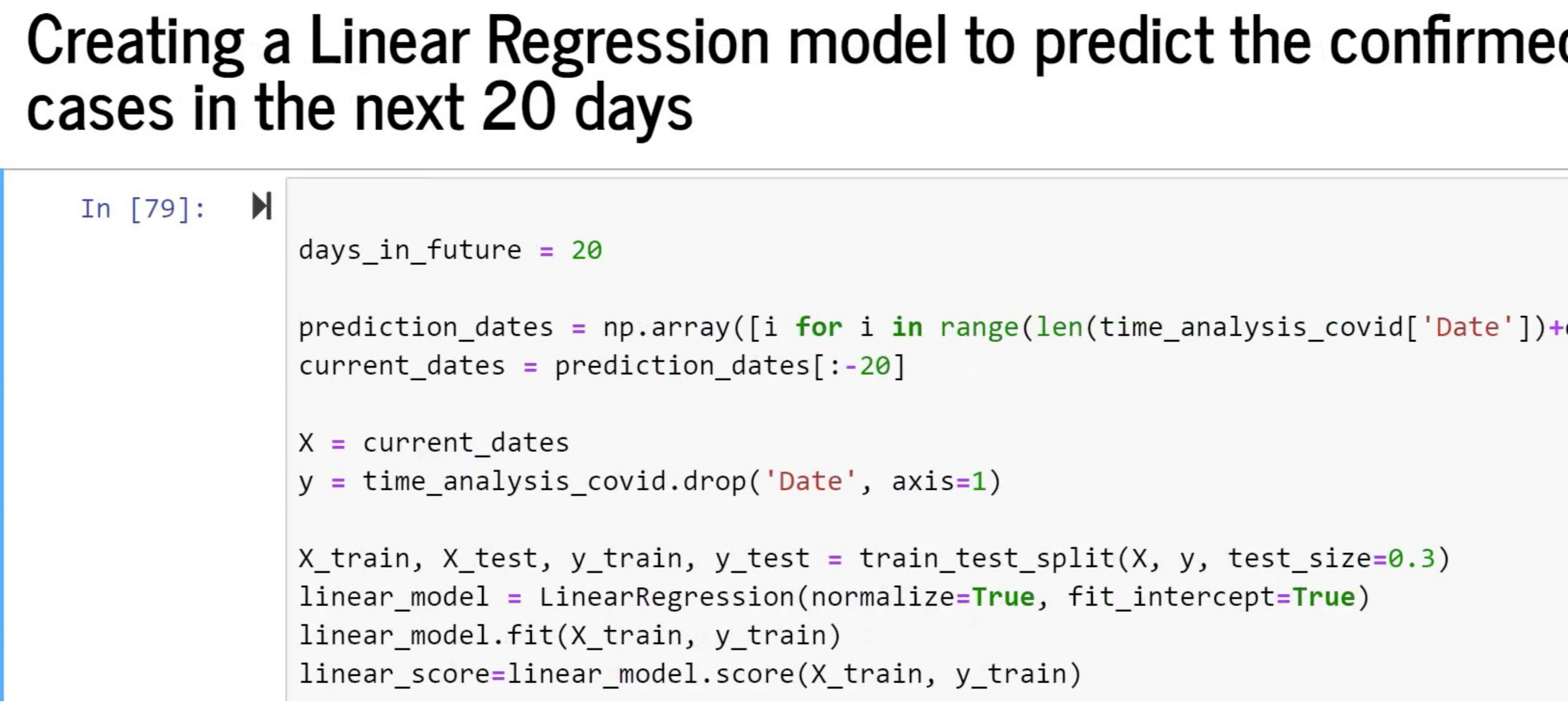
4. Covid-19 vs. SARS death rate comparison



Conclusions

- Even though covid-19 is spreading at a higher rate, there is a higher chance of recovering in comparison to SARS which has a higher death rate

Step 4. Data Modeling



Creating a Linear Regression model to predict the confirme cases in the next 20 days

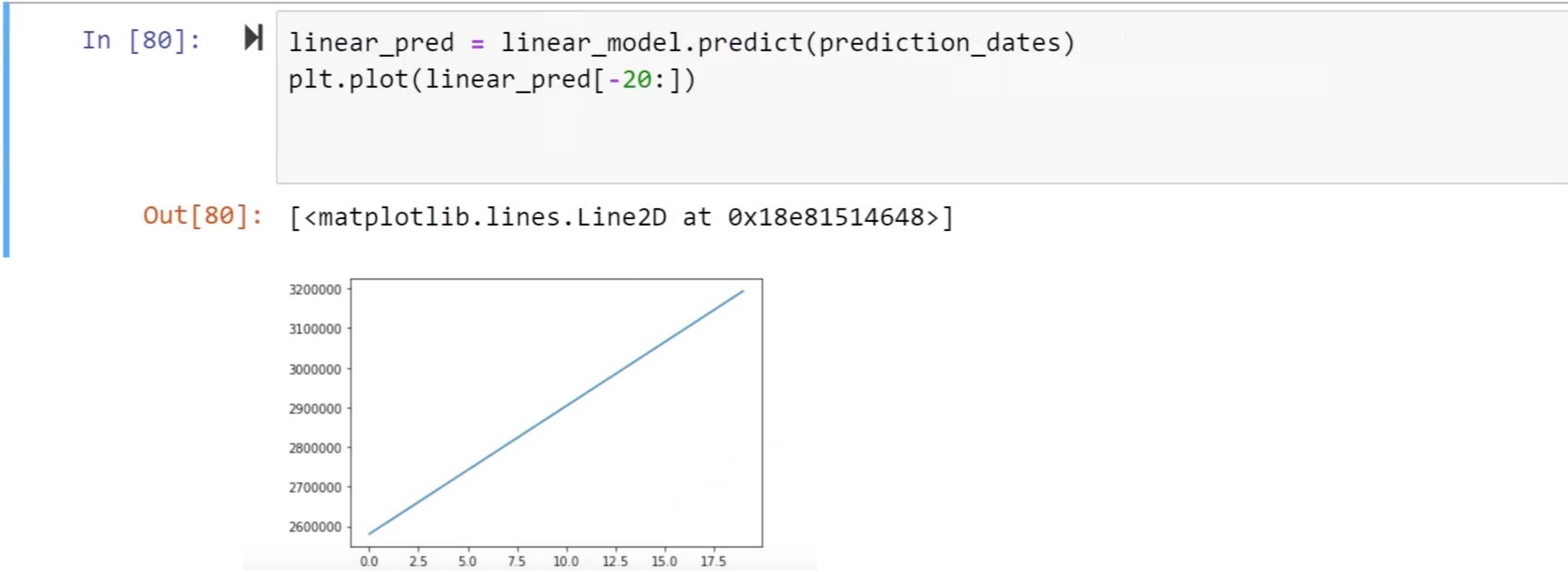
In [79]:

```
days_in_future = 20

prediction_dates = np.array([i for i in range(len(time_analysis_covid['Date'])+
current_dates = prediction_dates[:-20]

X = current_dates
y = time_analysis_covid.drop('Date', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
linear_model = LinearRegression(normalize=True, fit_intercept=True)
linear_model.fit(X_train, y_train)
linear_score=linear_model.score(X_train, y_train)
```

Linear Regression Model

- Our linear regression model has a score of **0.76684**

Graphing the linear regression prediction for the next 20 days



```
In [80]:  linear_pred = linear_model.predict(prediction_dates)
          plt.plot(linear_pred[-20:])
```

```
Out[80]: [<matplotlib.lines.Line2D at 0x18e81514648>]
```

Conclusion

Our model predicts that the confirmed cases will conitnue to rise steadily. However, a 75% score is not that good and might affect the accuracy of the data.

Challenges

- There was no data available regarding the SARS recovery rates. To proceed, I used the assumption that out of the confirmed cases, those who did not die recovered from the virus.

THE END