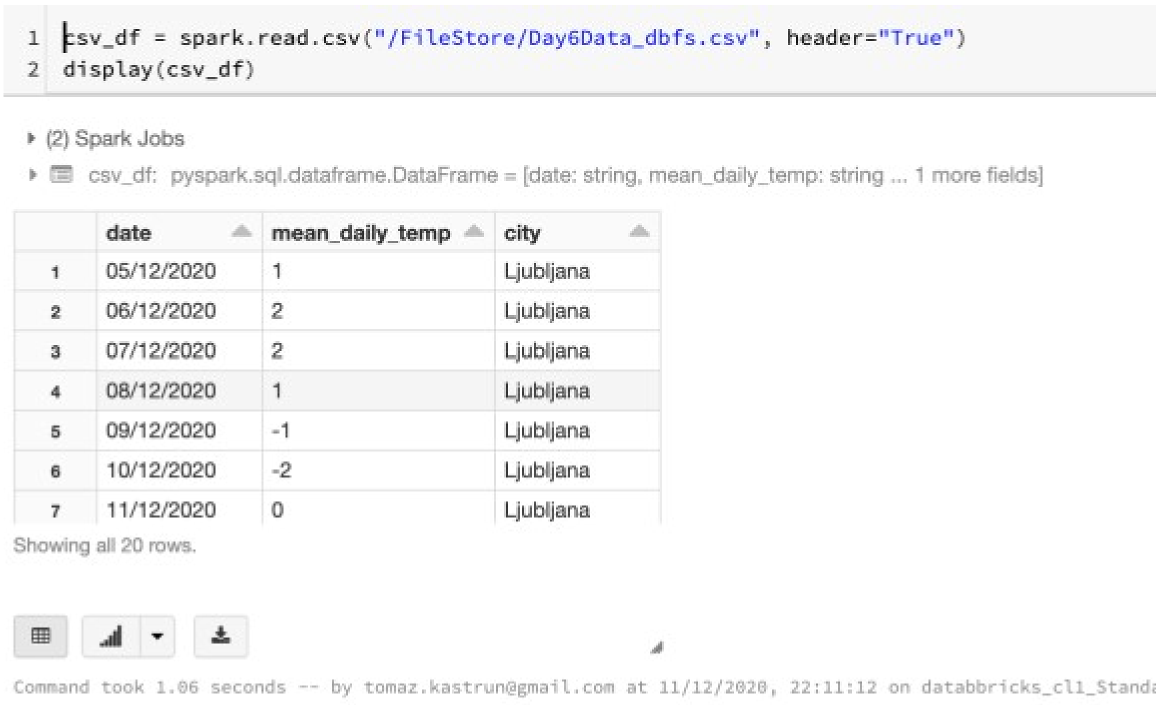
```
csv_df = spark.read.csv("/FileStore/Day6Data_dbfs.csv", header="True")
display(csv_df)
```



```
csv_df = spark.read.csv("/FileStore/Day6Data_dbfs.csv", header="True")
display(csv_df)
```

▸ (2) Spark Jobs
▸ csv_df: pyspark.sql.dataframe.DataFrame = [date: string, mean_daily_temp: string ... 1 more fields]

| | date | mean_daily_temp | city |
|---|---|---|---|
| 1 | 05/12/2020 | 1 | Ljubljana |
| 2 | 06/12/2020 | 2 | Ljubljana |
| 3 | 07/12/2020 | 2 | Ljubljana |
| 4 | 08/12/2020 | 1 | Ljubljana |
| 5 | 09/12/2020 | -1 | Ljubljana |
| 6 | 10/12/2020 | -2 | Ljubljana |
| 7 | 11/12/2020 | 0 | Ljubljana |

Showing all 20 rows.

Command took 1.06 seconds -- by tomaz.kastrun@gmail.com at 11/12/2020, 22:11:12 on databbricks_cl1_Standa

We can also import data from SQL Table into data frame by simply writing an SQL statement.

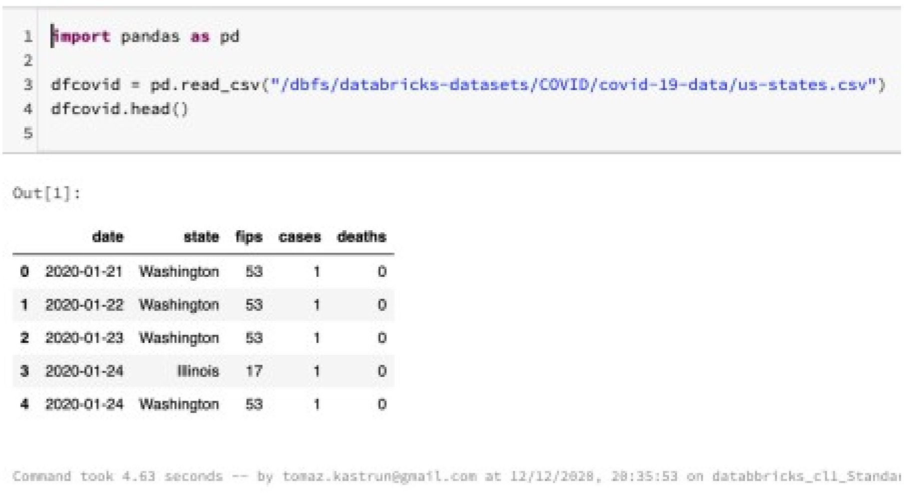
```
#from pyspark.sql.functions import explode
from pyspark.sql import *
import pandas as pd

display(sql("select * from day10.temperature"))
```

Besides displaying dataset, you can store a result of a query to a variable and use it later.

```
#for display
display(sql("select * from day10.temperature"))

#to save to variable
df = sql("select * from day10.temperature")
```

Let’s get now some data from Databricks sample data (that is available to anybody). So you can insert data from dbfs store and use the sample datasets as well, by using Python Pandas.

```
import pandas as pd

dfcovid = pd.read_csv("/dbfs/databricks-datasets/COVID/covid-19-data/us-
states.csv")
dfcovid.head()
```



```
import pandas as pd

dfcovid = pd.read_csv("/dbfs/databricks-datasets/COVID/covid-19-data/us-states.csv")
dfcovid.head()

```

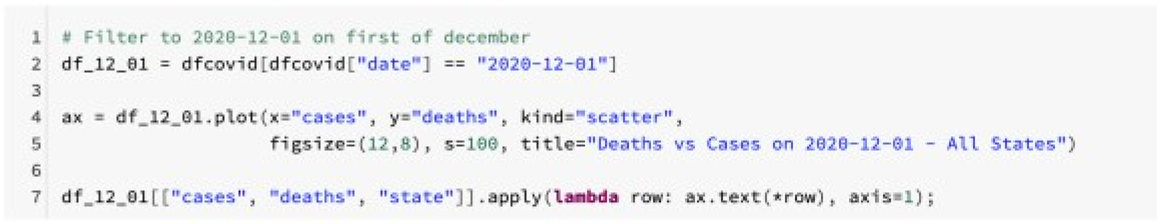
Out[1]:

|   | date | state | fips | cases | deaths |
|---|---|---|---|---|---|
| 0 | 2020-01-21 | Washington | 53 | 1 | 0 |
| 1 | 2020-01-22 | Washington | 53 | 1 | 0 |
| 2 | 2020-01-23 | Washington | 53 | 1 | 0 |
| 3 | 2020-01-24 | Illinois | 17 | 1 | 0 |
| 4 | 2020-01-24 | Washington | 53 | 1 | 0 |

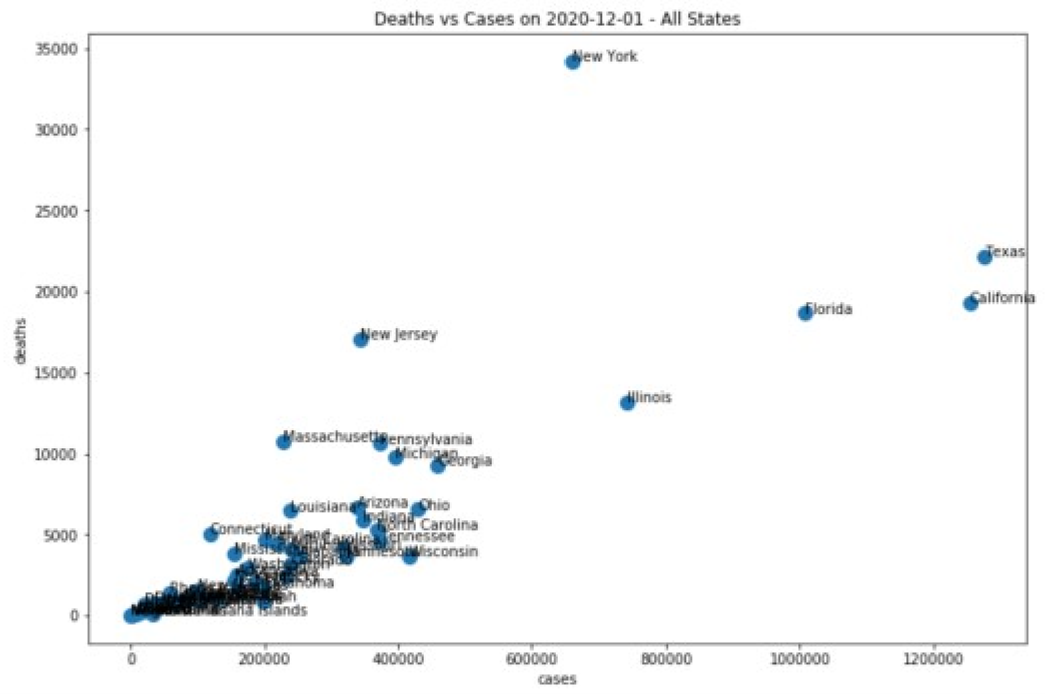
```
Command took 4.63 seconds -- by tomaz.kastrun@gmail.com at 12/12/2020, 20:35:53 on databbricks_cl1_Standa
```

and now let's scatter plot some number of cases and deaths per states and use the following Python cod
that can be simply used in Azure Databricks.

```
# Filter to 2020-12-01 on first of december
df_12_01 = dfcovid[dfcovid["date"] == "2020-12-01"]

ax = df_12_01.plot(x="cases", y="deaths", kind="scatter",
                   figsize=(12,8), s=100, title="Deaths vs Cases on 2020-12-0
- All States")

df_12_01[["cases", "deaths", "state"]].apply(lambda row: ax.text(*row),
axis=1);
```



```
# Filter to 2020-12-01 on first of december
df_12_01 = dfcovid[dfcovid["date"] == "2020-12-01"]

ax = df_12_01.plot(x="cases", y="deaths", kind="scatter",
                   figsize=(12,8), s=100, title="Deaths vs Cases on 2020-12-01 - All States")

df_12_01[["cases", "deaths", "state"]].apply(lambda row: ax.text(*row), axis=1);
```

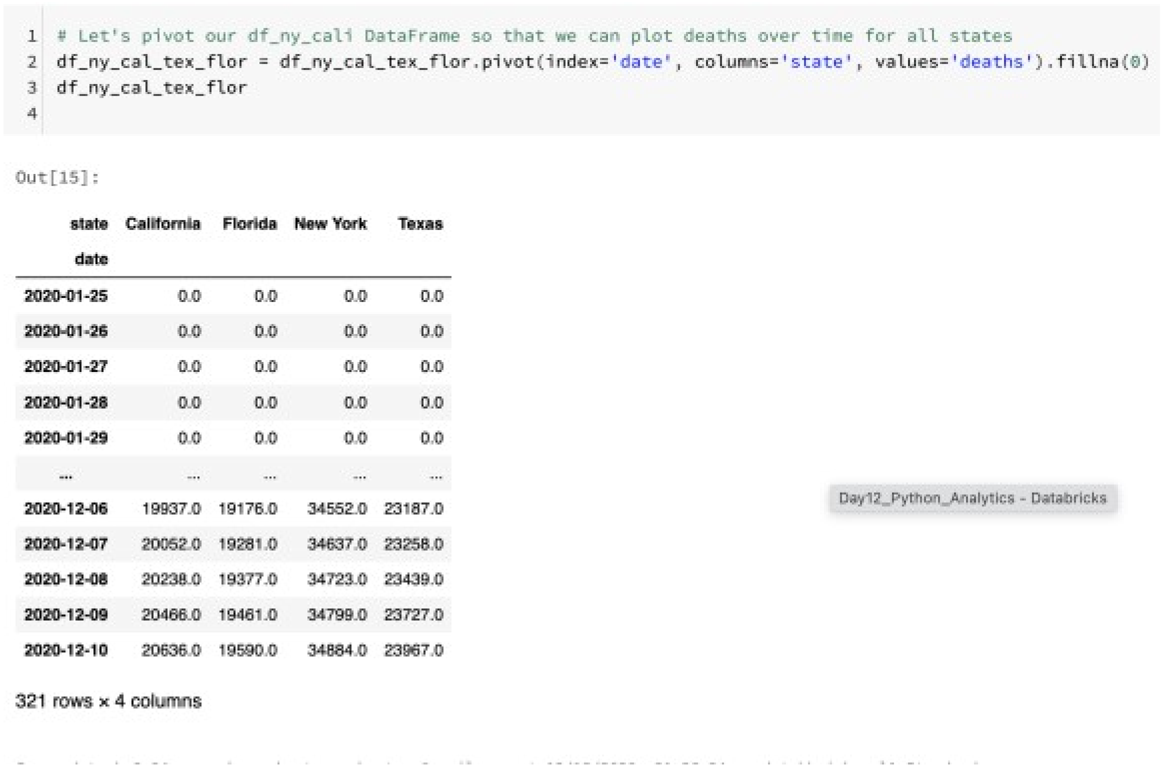


And now let’s compare only couple of these extreme states (New York, Texas, California and Florida). An create a subset for only these four states:

```
df_ny_cal_tex_flor = dfcovid[(dfcovid["state"] == "New York") |
(dfcovid["state"] == "California") | (dfcovid["state"] == "Florida") |
(dfcovid["state"] == "Texas")]
```

And now to create an index for the plot of deaths over time

```
df_ny_cal_tex_flor = df_ny_cal_tex_flor.pivot(index='date', columns='state',
values='deaths').fillna(0)
df_ny_cal_tex_flor
```

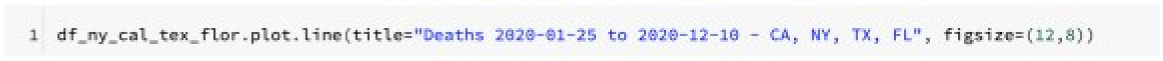


```
# Let's pivot our df_ny_cali DataFrame so that we can plot deaths over time for all states
df_ny_cal_tex_flor = df_ny_cal_tex_flor.pivot(index='date', columns='state', values='deaths').fillna(0)
df_ny_cal_tex_flor
```

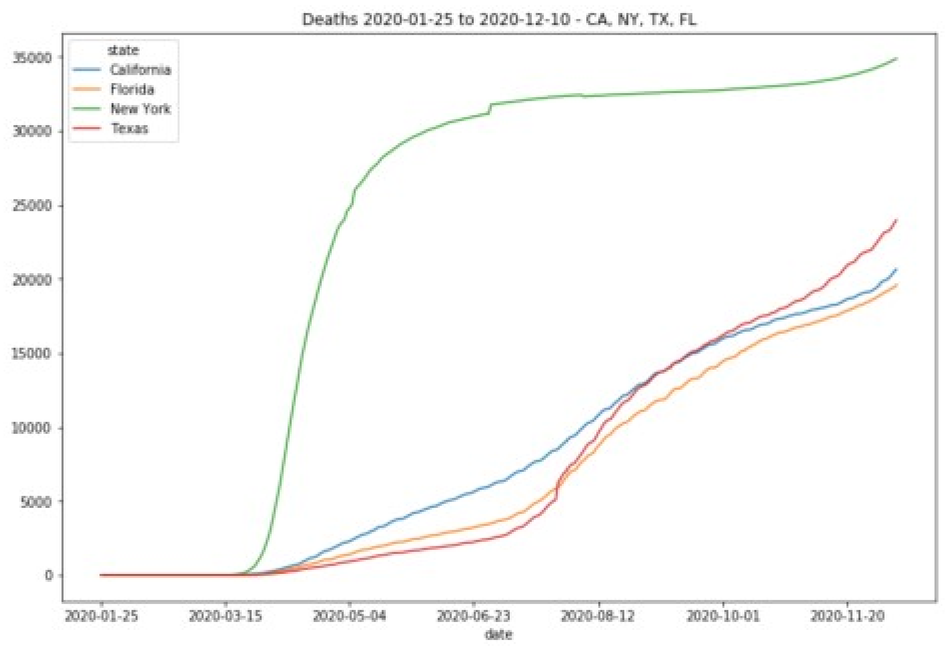
```
Out[15]:

     state  California  Florida  New York    Texas
      date
2020-01-25         0.0      0.0       0.0      0.0
2020-01-26         0.0      0.0       0.0      0.0
2020-01-27         0.0      0.0       0.0      0.0
2020-01-28         0.0      0.0       0.0      0.0
2020-01-29         0.0      0.0       0.0      0.0
       ...         ...      ...       ...      ...
2020-12-06     19937.0  19176.0   34552.0  23187.0
2020-12-07     20052.0  19281.0   34637.0  23258.0
2020-12-08     20238.0  19377.0   34723.0  23439.0
2020-12-09     20466.0  19461.0   34799.0  23727.0
2020-12-10     20636.0  19590.0   34884.0  23967.0

321 rows × 4 columns
```

and now plot this data using this dataset:

```
df_ny_cal_tex_flor.plot.line(title="Deaths 2020-01-25 to 2020-12-10 - CA, NY,
TX, FL", figsize=(12,8))
```



```
df_ny_cal_tex_flor.plot.line(title="Deaths 2020-01-25 to 2020-12-10 - CA, NY, TX, FL", figsize=(12,8))
```



And now for a simple regression analysis, we will split data from test and train. Since the first and second wave we will need to thing how to split the data. Let's split it until mid of November and after mid of November.

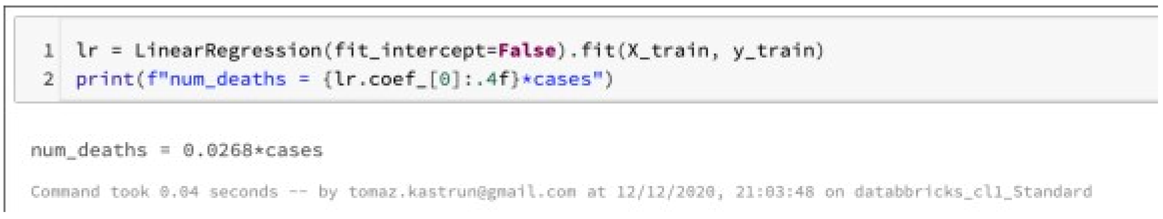
```
train_df = dfcovid[(dfcovid["date"] >= "2020-07-01") & (dfcovid["date"] <=
"2020-11-15")]
test_df = dfcovid[dfcovid["date"] > "2020-11-16"]

X_train = train_df[["cases"]]
y_train = train_df["deaths"]

X_test = test_df[["cases"]]
y_test = test_df["deaths"]
```

We will use scikit-learn to do simple linear regression.

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression().fit(X_train, y_train)
print(f"num_deaths = {lr.intercept_:.4f} + {lr.coef_[0]:.4f}*cases")
```

So if we have no cases, then there should be no deaths caused by COVID-19; this gives us a base line and assume that let's set the intercept to be 0.

```
lr = LinearRegression(fit_intercept=False).fit(X_train, y_train)
print(f"num_deaths = {lr.coef_[0]:.4f}*cases")
```



```
lr = LinearRegression(fit_intercept=False).fit(X_train, y_train)
print(f"num_deaths = {lr.coef_[0]:.4f}*cases")

num_deaths = 0.0268*cases

Command took 0.04 seconds -- by tomaz.kastrun@gmail.com at 12/12/2020, 21:03:48 on databbricks_cl1_Standard
```

This model imposes that there is a 2.68% mortality rate in our dataset. But we know that some states have higher mortality rates and that linear model is absolutely not ideal for that, but it is just to showcase for using Python in Databricks.