# 1. Basic Functions

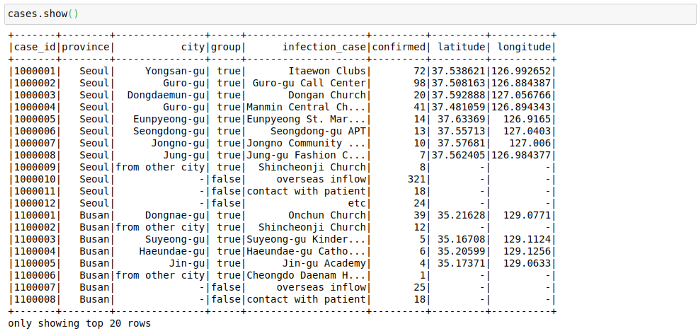
## Read

We can start by loading the files in our dataset using the spark.read.load command. This command reads parquet files, which is the default file format for spark, but you can add the parameter format to read .csv files using it.

cases = spark.read.load("/home/rahul/projects/sparkdf/coronavirusdataset/Case.csv",format="csv", sep=",", inferSchema="true", header="true")

## See a few rows in the file

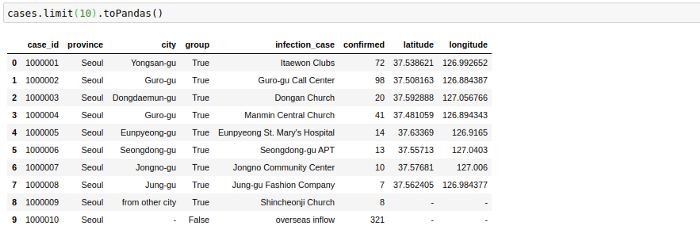
cases.show()



This file contains the cases grouped by way of the infection spread. This might have helped in the rigorous tracking of Corona Cases in South Korea.

The way this file looks is great right now, but sometimes as we increase the number of columns, the formatting becomes not too great. I have noticed that the following trick helps in displaying in pandas format in my Jupyter Notebook. The .toPandas() function converts a spark dataframe into a pandas Dataframe which is easier to show.

cases.limit(10).toPandas()



## Change Column Names

Sometimes we would like to change the name of columns in our Spark Dataframes. We can do this simply using the below command to change a single column:

cases = cases.withColumnRenamed("infection\_case","infection\_source")

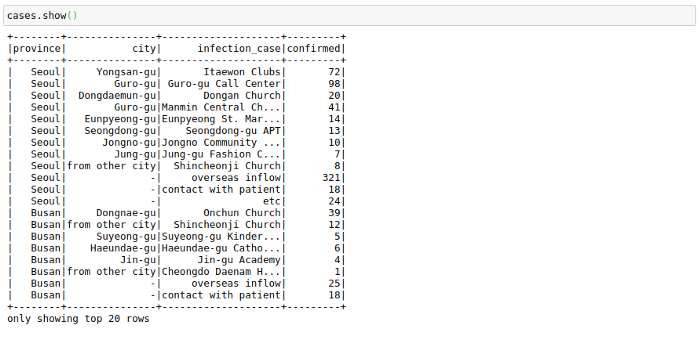
Or for all columns:

cases = cases.toDF(\*['case\_id', 'province', 'city', 'group', 'infection\_case', 'confirmed',  
 'latitude', 'longitude'])

## Select Columns

We can select a subset of columns using the select keyword.

cases = cases.select('province','city','infection\_case','confirmed')  
cases.show()



## Sort

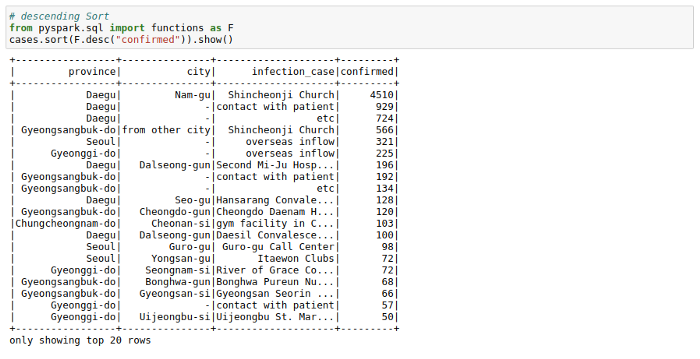
We can sort by the number of confirmed cases. Here note that the cases data frame will not change after performing this command as we don’t assign it to any variable.

cases.sort("confirmed").show()



But that is inverted. We want to see the most cases at the top. We can do this using the F.desc function:

# descending Sort  
from pyspark.sql import functions as F  
cases.sort(F.desc("confirmed")).show()



We can see the most cases in a logical area in South Korea originated from `Shincheonji Church`.

## Cast

Though we don’t face it in this dataset, there might be scenarios where Pyspark reads a double as integer or string, In such cases, you can use the cast function to convert types.

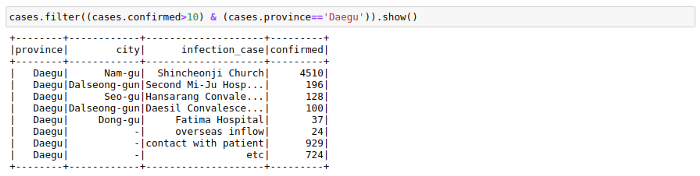
from pyspark.sql.types import DoubleType, IntegerType, StringTypecases = cases.withColumn('confirmed', F.col('confirmed').cast(IntegerType()))

cases = cases.withColumn('city', F.col('city').cast(StringType()))

## Filter

We can filter a data frame using multiple conditions using AND(&), OR(|) and NOT(~) conditions. For example, we may want to find out all the different infection\_case in Daegu Province with more than 10 confirmed cases.

cases.filter((cases.confirmed>10) & (cases.province=='Daegu')).show()



## GroupBy