Q1. Rationale for this choice:

Apache Spark is a high- performance framework used for processing, analyzing and making queries on Big data by using in-memory primitives. It it is therefore multiple times faster than various other programs used for big Data analysis and can provide real-time analysis.  
 - distributing the workload across a cluster of machines  
 -high-performance distributed computing framework  
-Apache Spark is a framework used for performing fast distributed computing on Big Data by using in-memory primitives.  
-It allows user programs to load data into memory and query it repeatedly, making it a well suited tool for online and iterative processing

**Spark setup:** Virtual box download, OS: Ubuntu:18.04, Creating -VM master. 1024MB memory size for VM, installing java and python by running python 3. Installing Spark 2.4.6 version with pre built Apache Hadoop2.7 and moving spark to /usr/local/folder. Changing .bashrc file to add the paths for Spark and python (**FIRST AND SECOND TUTORIAL)**  
  
-Downloading ‘covid19.csv’ to Downloads folder and moving it to spark folder by running commands in terminal: cd Downloads ,sudo mv covid19.csv /usr/local/spark  
  
-Use commands: cd /usr/local/spark and then run pyspark (so has access to all files in usr/local/spark , where ‘covid19’ has been moved )

Q2.  
To load, and read csv format and creating data frame simultaneously spark.read.format( “csv”) function was used. Ensure the inferSchema is set to True such that not all values are strings (e.g. numerical values in new\_cases, total\_cases, new\_deaths and total\_deaths)would cause issues while using functions such as max, filter etc) and can be sorted according to requirements stated in the coursework description. It could be achieved also using SparkContext (sc=spark.SparkContext) through sc.TextFile\* that reads the file and returns it as RDD. However, spark.read.format() returns a DataFrame, that was required for the purpose of this exercise. The filter function along with isNotNull() function was implemented to filter out any empty cells (null) from the data\_frame(e.g. where there were no values entered   
GroupBy() function in conjugation with aggregate function and sort() was used to firstly group all entries from the same countries/data by the country, find maximum value in the grouped data of total\_deaths and sort the results alphabetically by location  
Max function was used in conjugation with group by function, to find totalcases in each country, and then sort them in ascending and descending order  
  
\* code that converts RDD to Dataframe:   
sc= spark.sparkContext

df = sc.textFile(“covid19.csv”)

header = df.first()

data = df.filter(lambda x: x!=header).map(lambda line: line.split(","))

columns = header.split(",")

data\_frame = spark.createDataFrame(data, columns)

data\_frame.show()

cleanedData = data\_frame.na.drop() #removes all nulls

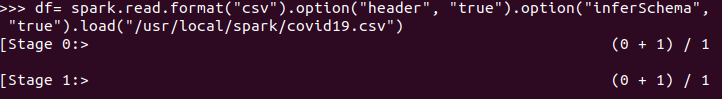
cleanedData.coalesce(1).write.option("header", True).option("delimiter", ",").csv('cleanedData')

Q3:  
Results: the code written for this task enable to find max and min numbers of total deaths and total cases grouped by country. According to data in the csv file as well as by running the code USA has highest total cases. The transformation functions such as filter, group by were applied to the data frame (in similar way as they would be to RDD if the date wasn’t converted into data frame the initial step).

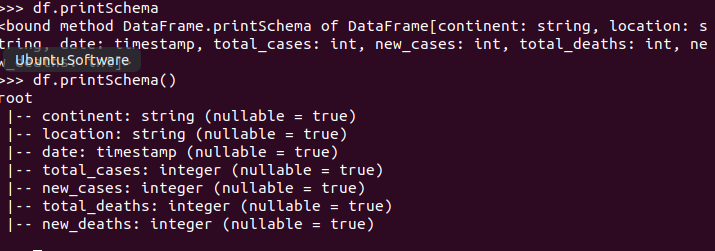
**In your report, comment on how this analysis could be extended to consider larger datasets (e.g., 4 Terabyte of COVID-19 for 2 years). Briefly Describe how to use your Spark skills to solve other problem (Chose your own case study):**

Program:   
1)loading

#Loading csv with headers set to true and simultaneously creating data frame (df) from the csv file instead of RDD’s .  
  
df = spark.read.format("csv").option("header", "true").option(“inferSchema”, “true”).option(“delimiter”, “,”).load("/usr/local/spark/covid19.csv")

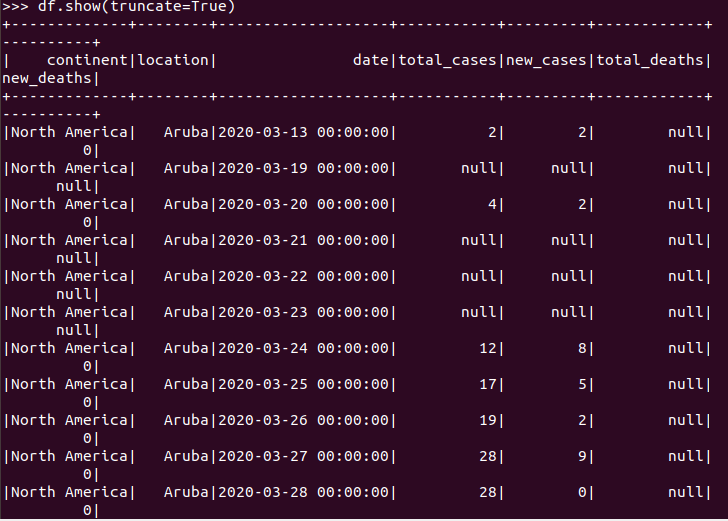


#printing schema (allows to see whether collums have been created correctly and whether contain correct data type, e.g. strings for places and integers for numerical values)

df.printSchema()  
 

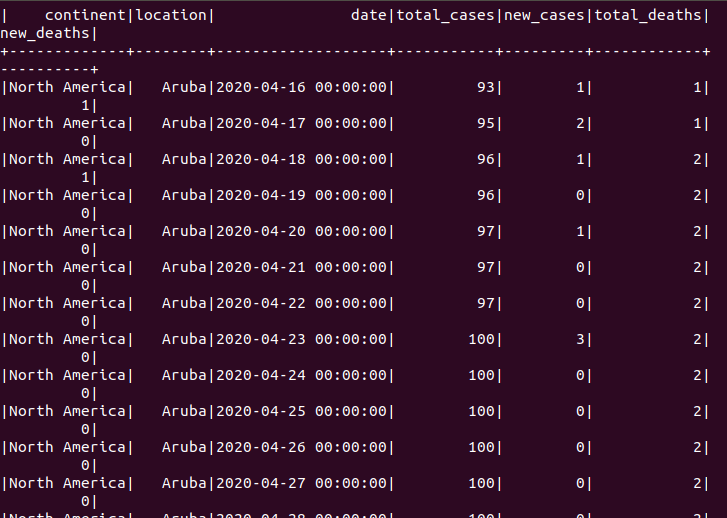
# show() command used to display entire data\_frame (first 20 values ordered alphabetically)

df.show(truncate=True) to display



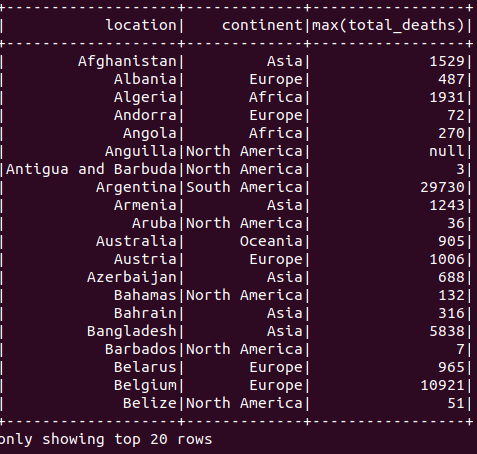
#filtering null values. IMPORTANT: 0s are not treated as nulls, only empty cells where there are no entries are regarded as nulls. If we wanted to filter both empty cells and 0, before filtering we could use replace() function, to replace all 0s with null and then filter using Filter() and isNotNull() functions, the same way as below:

df\_filtered=df.filter((df.new\_cases.isNotNull())&(df.new\_deaths.isNotNull())&(df.total\_deaths.isNotNull())&(df.total\_cases.isNotNull())).show(truncate=True)



#grouping by Location and continent, and seeing the highest total\_deaths per country using aggregate function, sorted alphabetically by name of the country

df\_grouped= df.groupBy([‘location’, ‘continent’]).agg({‘total\_deaths’:’max’}).sort(‘location’).show(truncate=False)



**# finding countries with highest number of total cases and sorting in descending order (first 20 shown)**

df\_max= df.groupBy([‘location’].max(‘total\_cases’).sort(‘max(total\_cases)’, ascending=False).show(truncate=true)



#Finding countries with lowest number of total\_cases, same as above, but setting ascending order to True. Using min() function would find the lowest value for each country in total\_cases column, which is cumulative, therefore wouldn’t show true number of total\_cases for each country, rather

df\_min= df.groupBy([‘location’].max(‘total\_cases’).sort(‘max(total\_cases)’, ascending=True).show(truncate=true)

