23/12/2023

Chakradhar Bhogapurapu,BIBA

Pyspark

Basic Spark Commands

🡪To start spark shell

Spark-shell

🡪Read the file from local system

Val data = sc.textFile(“data.txt”)

“sc” is the spark context.

🡪Create RDD through parallelizing

Val num = Array(1,2,3,4,5,6,7,8,9,10)

Val NewData = sc.parallelize(num)

🡪Count items in RDD

NewData.count()

🡪To read the first 5 items from RDD

NewData.take()

🡪Save output/processes data into the text file

Counts.saveAsextFile(“output”)

Here “Output” folder is the current path.

Intermediate spark commands

🡪Filter on RDD

Let’s create a new RDD for items that contain “yes”.

Val DFData = data.filter(line=>line.contains(“yes”))

To apply the transformation filter to an existing RDD and generate a new listof items, you must filter for the word “yes”.

🡪Chain operation

Data.filter(line=>line.contains(“DataFlair”)).count()

🡪Create RDD through parallelizing

Val num = Array(1,2,3,4,5,6,7,8,9,10)

Val NewData = sc.parallelize(num)

NewData is the RDD now

🡪Count RDD partitions

NewData.partitions.length

🡪Cache a file

data.cache()

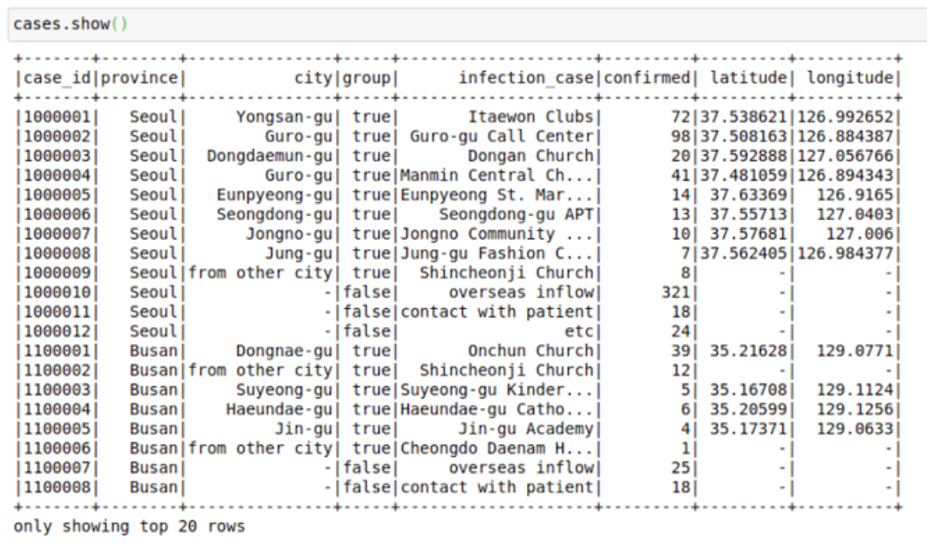
Functions of spark

🡪Read

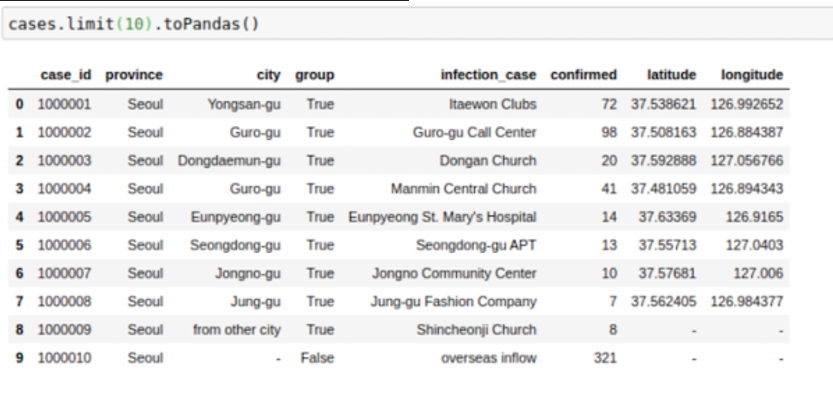
cases = spark.read.load(“path”, format=”csv”, sep = “,”), inferSchema = “true”, header=”true”

🡪See rows in the file

cases.show()



Cases.limit(10).toPandas()



🡪Change column names

To change name of columns in spark dataframes

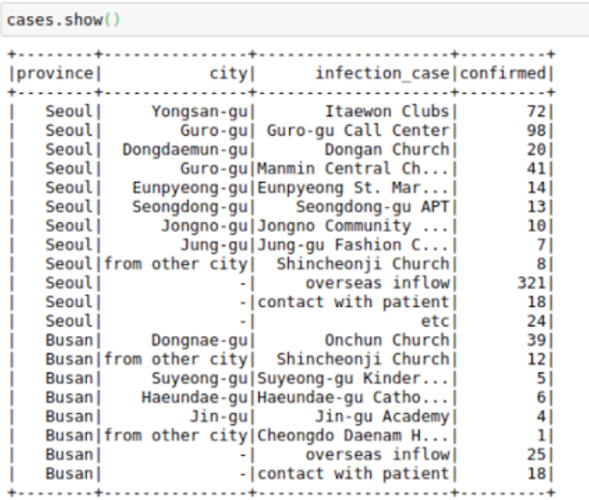
cases = cases.withColumnRenamed(“infection\_case”,”infection\_source”)

or for all columns:

cases = cases.toDF(\*[‘case\_id’,’province’,’city’,’group’,’infection\_case’,’longitude’])

🡪Select columns

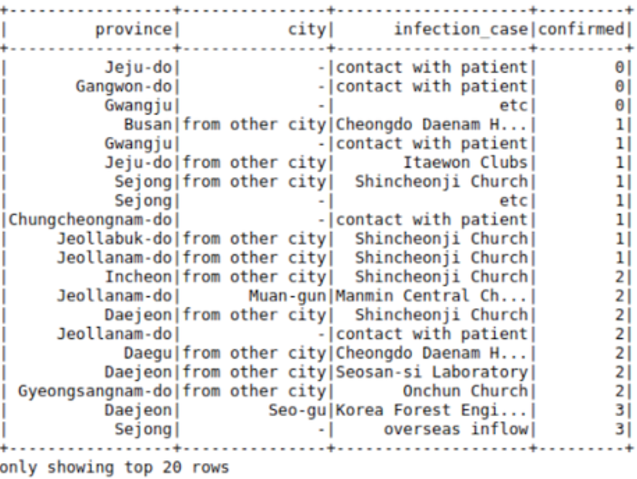
cases = cases.select(‘province’,’city’,’infection\_case’,’confirmed’)



cases.show() --- > this will show the entire table

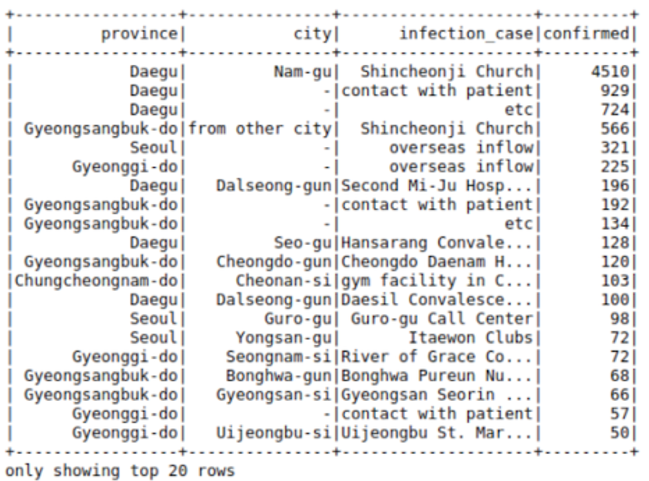
🡪Sort

Cases.sort(“confirmed”).show()



From pyspark.sql import functions as F

cases.sort(F.desc(“confirmed”)).show()



🡪Cast

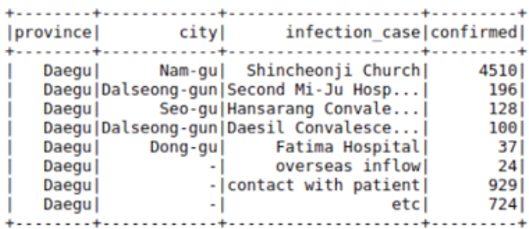
From pyspark.sql.types import DoubleType, IntegerType, StringType

cases = cases.withColumn(‘confirmed’, F.col(‘confirmed’).cast(IntegerType()))

cases = cases.withColumn(‘city’, F.col(‘city’).cast(Stringtype()))

🡪Filter

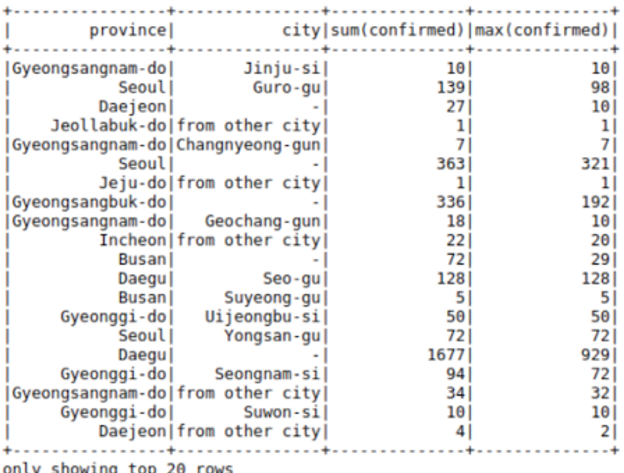
cases.filter((cases.confirmed>10)&(cases.province==’Daegu’)).show()



🡪Groupby

From pyspark.sql import funcitons as F

cases.groupBy((“province”,”city”)).agg(F.sum(“confirmed”),F.max(“confirmed”)).show()



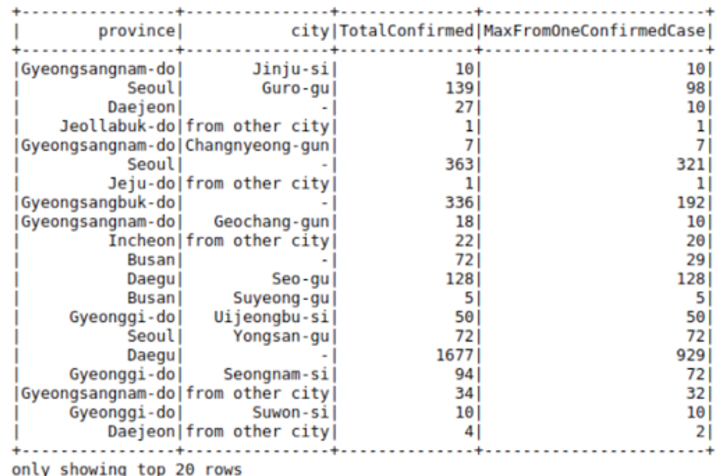
If we don’t like the new columns, we can use alias keyword to rename columns in the command itself

cases.groupBy([“province”,”city”]).agg(

F.sum(“confirmed”).alias(“TotalConfirmed”),\

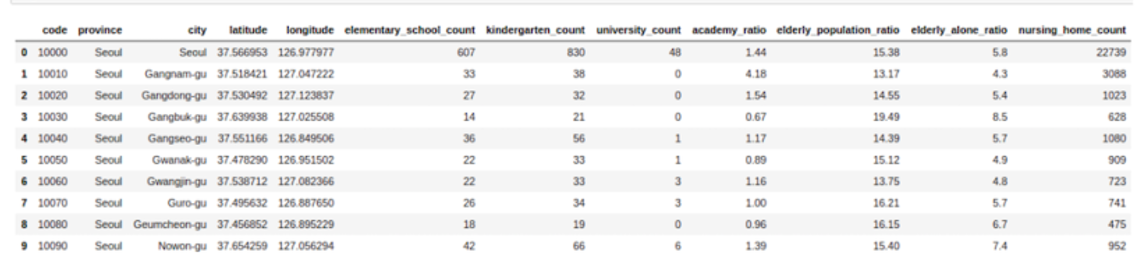
F.max(“confirmed”).alais(“MaxFromOneConfirmedCase”)\

).show()



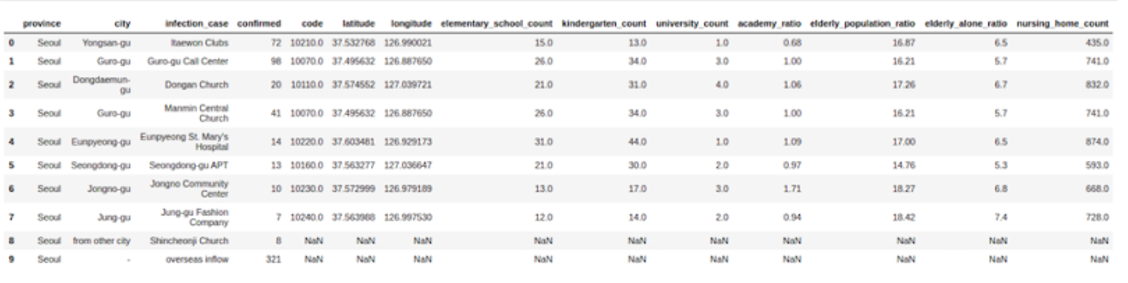
🡪Joins

regions = spark.read.load(“Path”,format=”csv”, sep=”,”, inferSchema=”true”, header=”true”)regions.limit(10).toPandas()

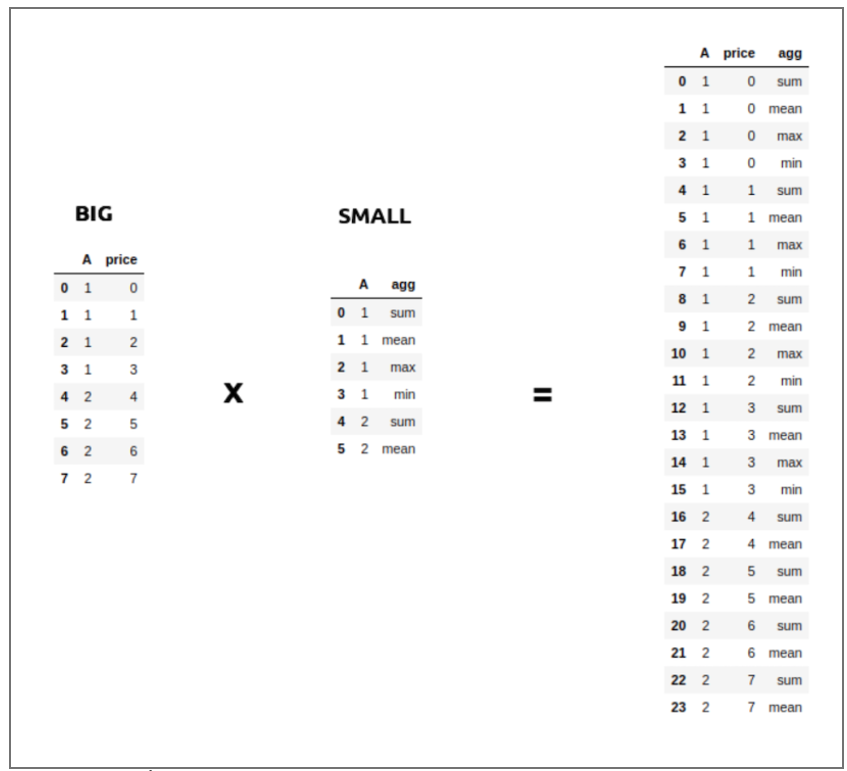


cases= cases.join(regions,[‘province’,’city’], how=’left’)

cases.limit(10).toPandas()



🡪Broadcast/Map side joins in pyspar Dataframes



from pyspark.sql.functions import broadcast

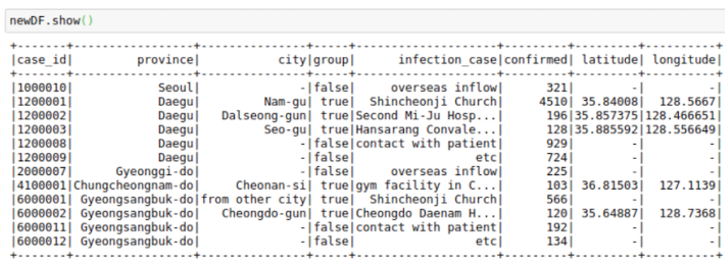
cases = cases.join(broadcast(regions),[‘province’,’city’], how = ‘left’)

🡪Use sql with pyspark dataframes

cases.registerTempTable(‘cases\_table’)

newDF = sqlContext.sql(‘select\* from cases\_table where confirmed>100’)

newDF.show()

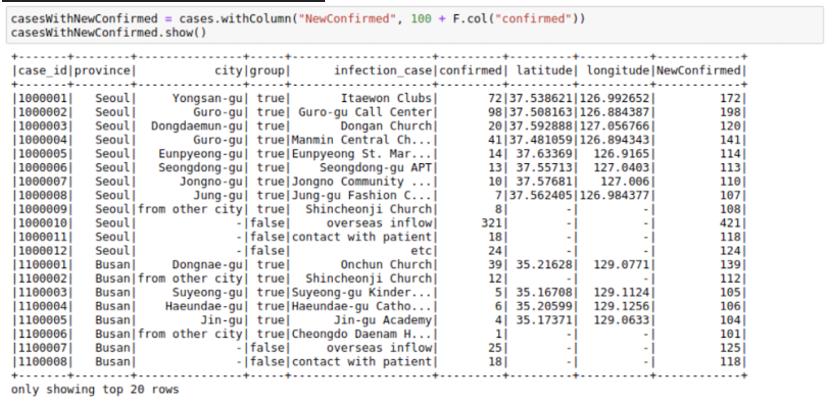


🡪Create new columns in pyspark dataframes  
using spark native commands

Import pyspark.sql.functions as F

casesWithNewConfirmed = cases.withColumn(“NewConfirmed”,100+F.col(“confirmed”))

casesWithNewConfirmed.show()



We can also use math funcitons like the F.exp function :

casesWithExpConfirmed = cases.withColumn(“ExpConfirmed”,F.exp(“cofirmed”))

casesWithExpConfirmed.show()

