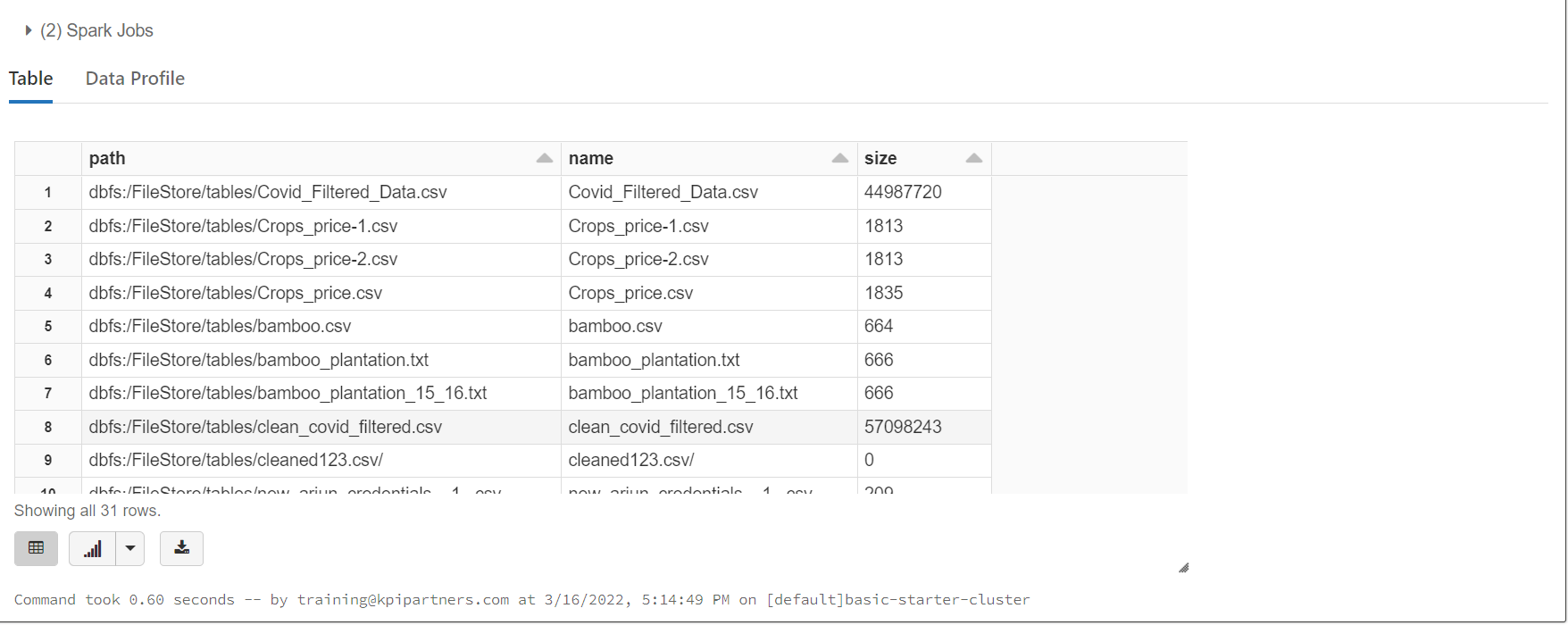
display(dbutils.fs.ls("/FileStore/tables"));

output :-



from pyspark import SparkContext

from pyspark import SparkConf

from pyspark.sql import SparkSession

from pyspark.sql.types import StructType, StructField, StringType, IntegerType,FloatType

import re

df= spark.read.csv("dbfs:/FileStore/tables/owid\_covid\_data.csv")

df.collect()

output :-



1.Replace all the negative values & null values with zero.

df1=df.fillna(0);

for i in range(len(df)):

if (df.at[i,'new\_cases'])<0:

df.at[i,'new\_cases']=0

if (df.at[i,'new\_cases\_smoothed'])<0:

df.at[i,'new\_cases\_smoothed']=0

if (df.at[i,'new\_deaths'])<0:

df.at[i,'new\_deaths']=0

if (df.at[i,'new\_deaths\_smoothed'])<0:

df.at[i,'new\_deaths\_smoothed']=0

if (df.at[i,'new\_cases\_per\_million'])<0:

df.at[i,'new\_cases\_per\_million']=0

if (df.at[i,'new\_cases\_smoothed\_per\_million'])<0:

df.at[i,'new\_cases\_smoothed\_per\_million']=0

if (df.at[i,'new\_deaths\_per\_million'])<0:

df.at[i,'new\_deaths\_per\_million']=0

if (df.at[i,'new\_deaths\_smoothed\_per\_million'])<0:

df.at[i,'new\_deaths\_smoothed\_per\_million']=0

if (df.at[i,'reproduction\_rate'])<0:

df.at[i,'reproduction\_rate']=0

if (df.at[i,'excess\_mortality\_cumulative\_per\_million'])<0:

df.at[i,'excess\_mortality\_cumulative\_per\_million']=0

if (df.at[i,'excess\_mortality'])<0:

df.at[i,'excess\_mortality']=0

if (df.at[i,'excess\_mortality\_cumulative'])<0:

df.at[i,'excess\_mortality\_cumulative']=0

if (df.at[i,'excess\_mortality\_cumulative\_absolute'])<0:

df.at[i,'excess\_mortality\_cumulative\_absolute']=0

if (df.at[i,'excess\_mortality\_cumulative'])<0:

df.at[i,'excess\_mortality\_cumulative']=0

2.remove the records where the continent column having blanks

for i in range(len(df)):

if (df.at[i,'continent'])==0 : # (pd.isnull(df.at[i,'continent']) or

df.drop([i], axis=0, inplace=True)

df1=spark.read.format("csv").option("header","true").load("dbfs:/FileStore/tables/Covid\_Filtered\_Data.csv")

df1.show();

df1.createOrReplaceTempView("Covid")

Using SQL :-

%sql

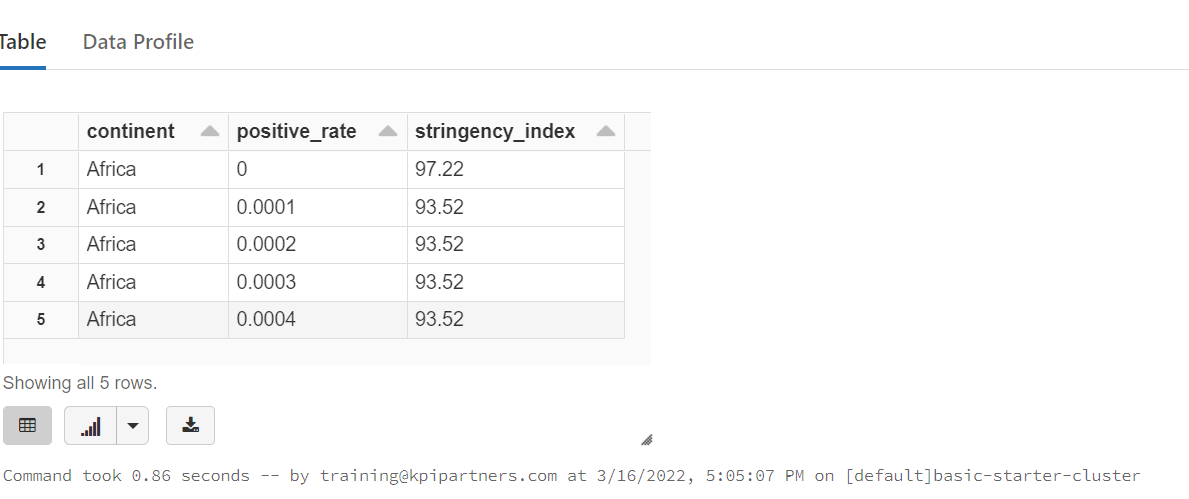
3. display the continent,positivity rate where the strigency index is maximum

select continent,positive\_rate, max(stringency\_index) as stringency\_index from Covid

group by continent,positive\_rate

limit 5;

output :-



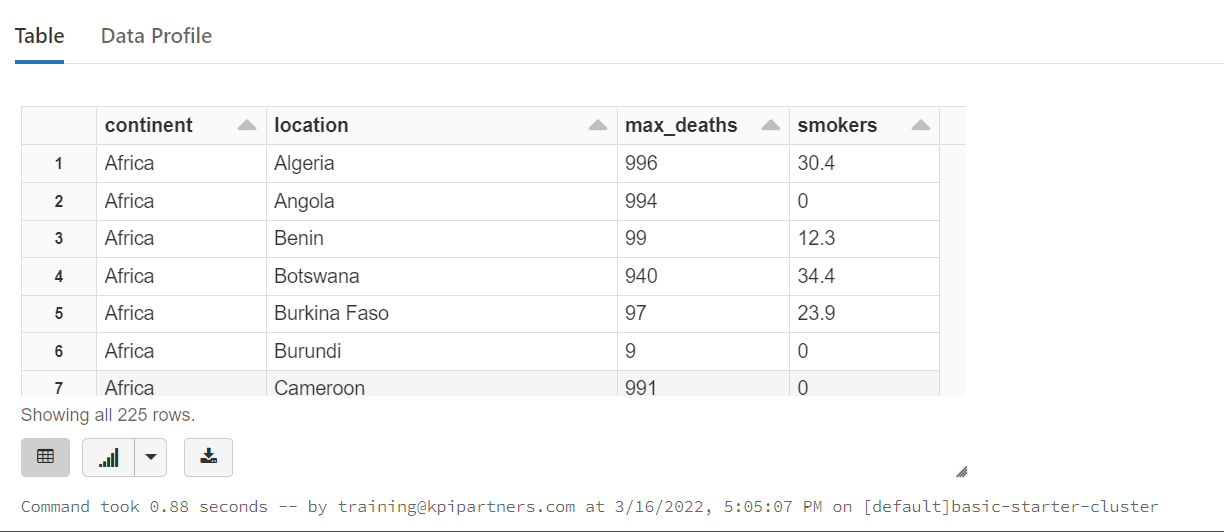
%sql

4. which continent and location having max of deaths and male\_smokers

select continent,location,max(total\_deaths) as max\_deaths ,max(male\_smokers)as smokers from Covid

group by continent,location;

output :-



5. which year having the highest number of deaths

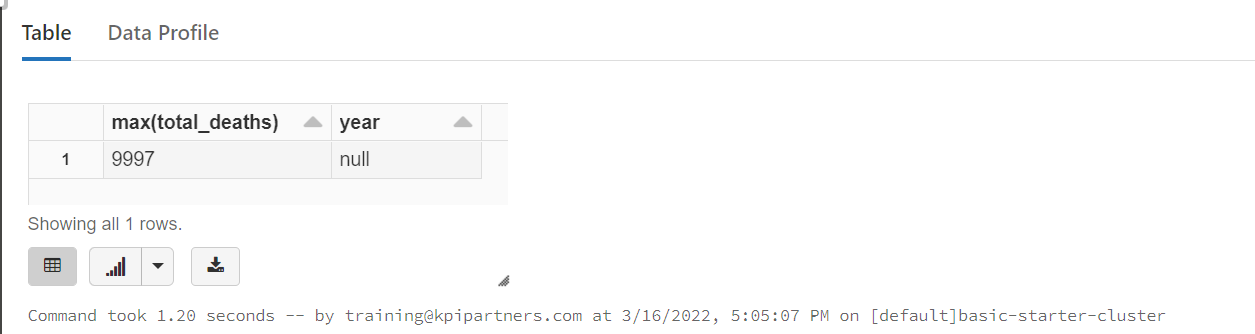
%sql

select max(total\_deaths),year(date) as year from Covid group by(year)

order by year

limit 10;

output :-



6. which location having lowest recovery rate for each continent

Using python :-

from pyspark import SparkContext

from pyspark import SparkConf

from pyspark.sql import SparkSession

from pyspark.sql.types import StructType, StructField, StringType, IntegerType,FloatType

import re

df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/tables/Covid\_Filtered\_Data.csv")

df1.createOrReplaceTempView("Covid")

df1.show()



df1=spark.sql("select first(location),first(total\_cases - total\_deaths) as recovery from Covid group by(continent) ").show()



7. Countries with max cardiovasc\_death\_rate and diabetes\_prevalence greater than 10

Using Scala :-

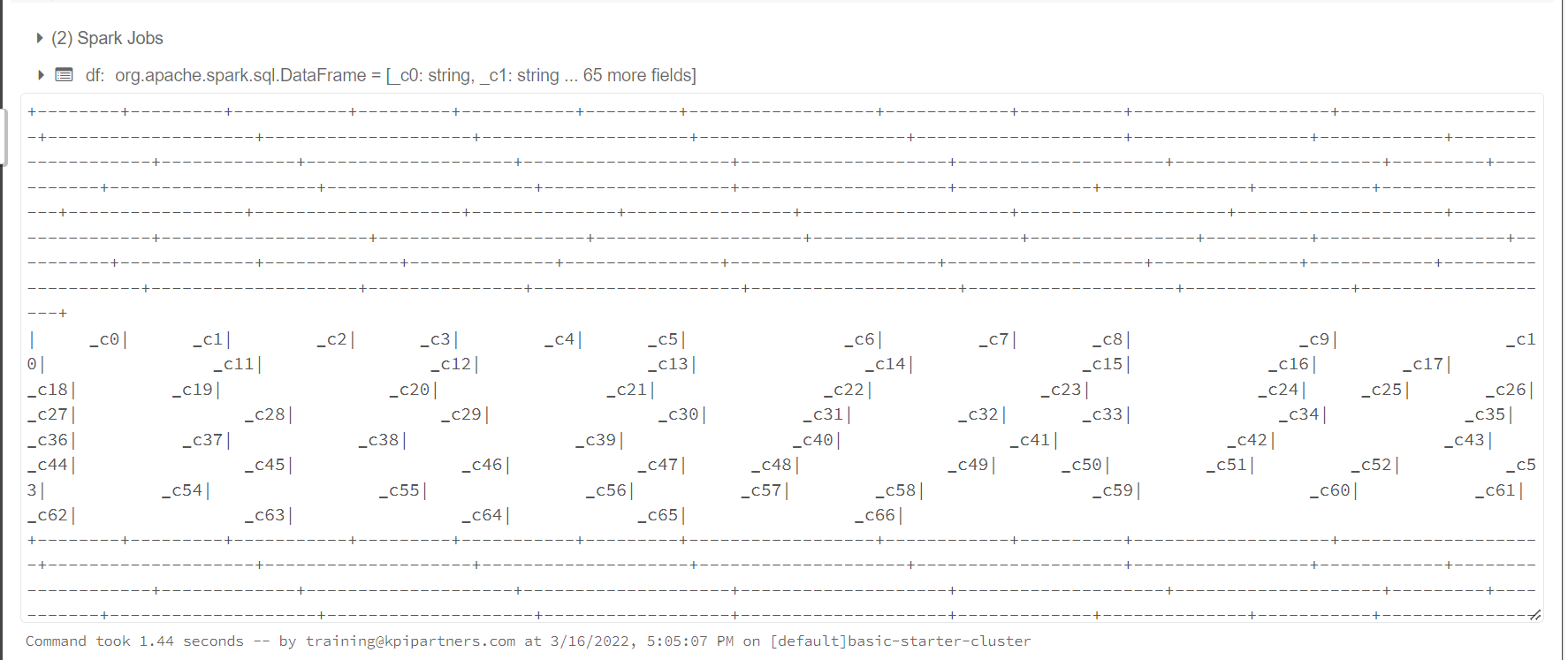
%scala

import org.apache.spark.sql.SparkSession

"""from scalaspark.sql.functions import \*"""

val df = spark.read.csv("dbfs:/FileStore/tables/Covid\_Filtered\_Data.csv")

df.show()

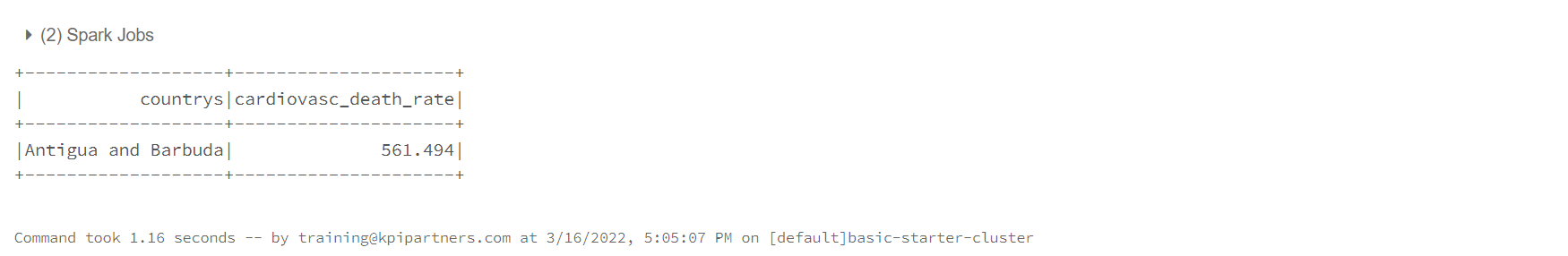


%scala

df.createOrReplaceTempView("covid")

%scala

spark.sql(" select first(\_c2) as countrys , max(\_c55) as cardiovasc\_death\_rate from covid where \_c56 >10 ;").show()



8. which location has highest number of vaccinated people in each continent

%scala

spark.sql("select first(\_c2),max(\_c35) as Vaccination from Covid group by \_c2 order by Vaccination").show();

output :-

