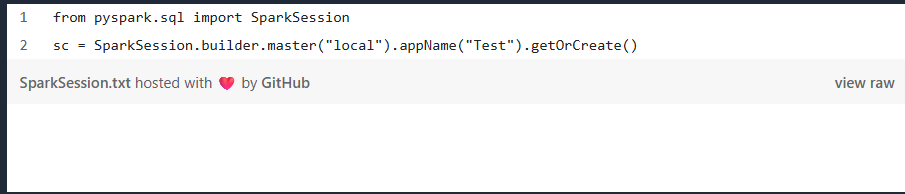
**Using PySpark for Dealing with Larger than Memory Datasets**

**Objective:**

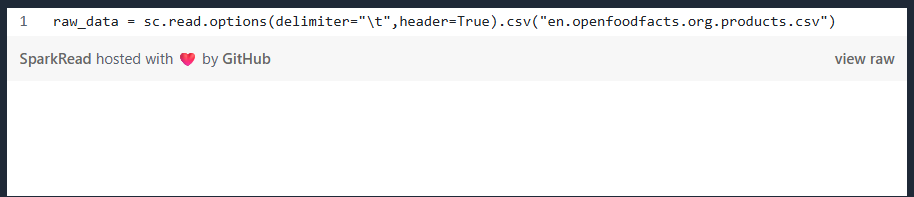
I created a 3GB VM to simulate working with datasets larger than available memory. I have a 4.2GB CSV file of Open Food Facts data that I need to perform exploratory data analysis (EDA) on, but I need to do this without causing memory crashes.

**I took the following steps in order to set up my environment on Ubuntu :**

1. Install Anaconda
2. Install Java openJDK 11: sudo apt-get install openjdk-11-jdk. The Java version is important as Spark only works with Java 8 or 11
3. Install Apache Spark and configure the Spark environment (add SPARK\_HOME variable to PATH). If all went well you should be able to launch spark-shell in your terminal
4. Install pyspark: conda install -c conda-forge pyspark

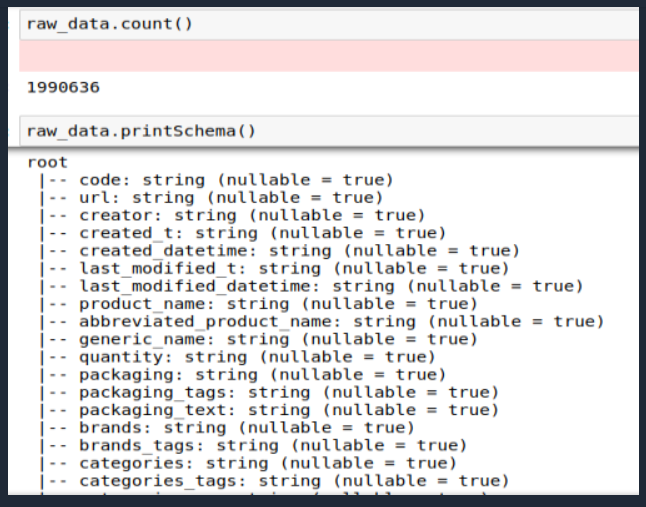


I read the data from my large csv file inside my SparkSession using *sc.read.* Trying to load a 4.2 GB file on a VM with only 3 GB of RAM does not issue any error as Spark does not actually attempt to read the data unless some type of computation is required.

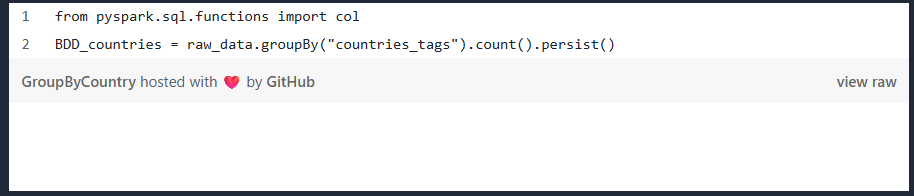


The result is a pyspark.sql.dataframe variable. It is important to keep in mind that, at this point, the data is not actually loaded into the RAM memory. Data is only loaded when an action is called on the pyspark data frame, an action that needs to return a computed value. If I ask for instance for a count of the number of products in the data set, Spark is smart enough not to try and load the whole 4.2 GB of data in order to compute this value (almost 2 million products).

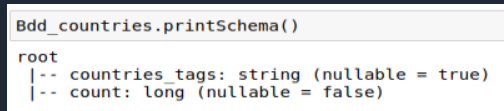
I start by using the printSchema function from pyspark in order to get some information about the structure of the data: the columns and their associated type :



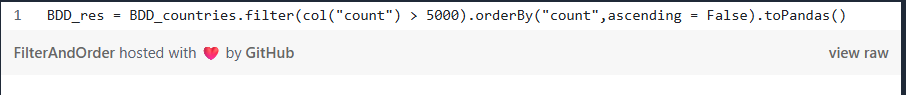
To start the exploratory analysis, I computed the number of products per country to get an idea of the database composition :

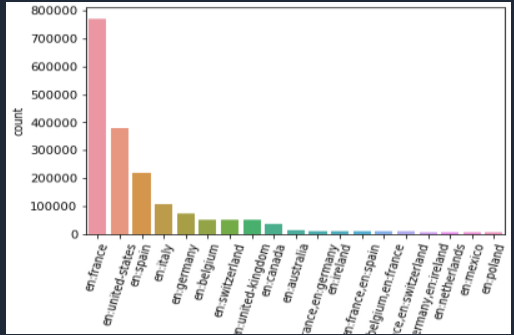


The result of this operation, BDD\_countries, is also a pyspark data frame and has the following structure :



I can filter this new data frame to keep only the countries that have at least 5000 products recorded in the database and plot the result :





**Summary:**

By reading the data using a Spark Session, it is possible to perform basic exploratory analysis computations without actually trying to load the complete data set into memory. This type of approach can be useful when we want to be able to get a first impression of the data and search for ways to identify and filter out unnecessary information.