Documentation – 19th May 2021

Tasks –

1) Write a Pyspark code for word count

2) Documenting the Understanding of the code and internals of how partitioning in RDD is done and Relation with JVM till its last cluster.

**Pyspark Code Word count-**

**ALGO ->**

1) Import the spark libraries (Spark session and context)

2) Initialize spark session through determining Builder, Master, appname and getOrCreate() method.

3)initialize the spark context

4) Import .txt file

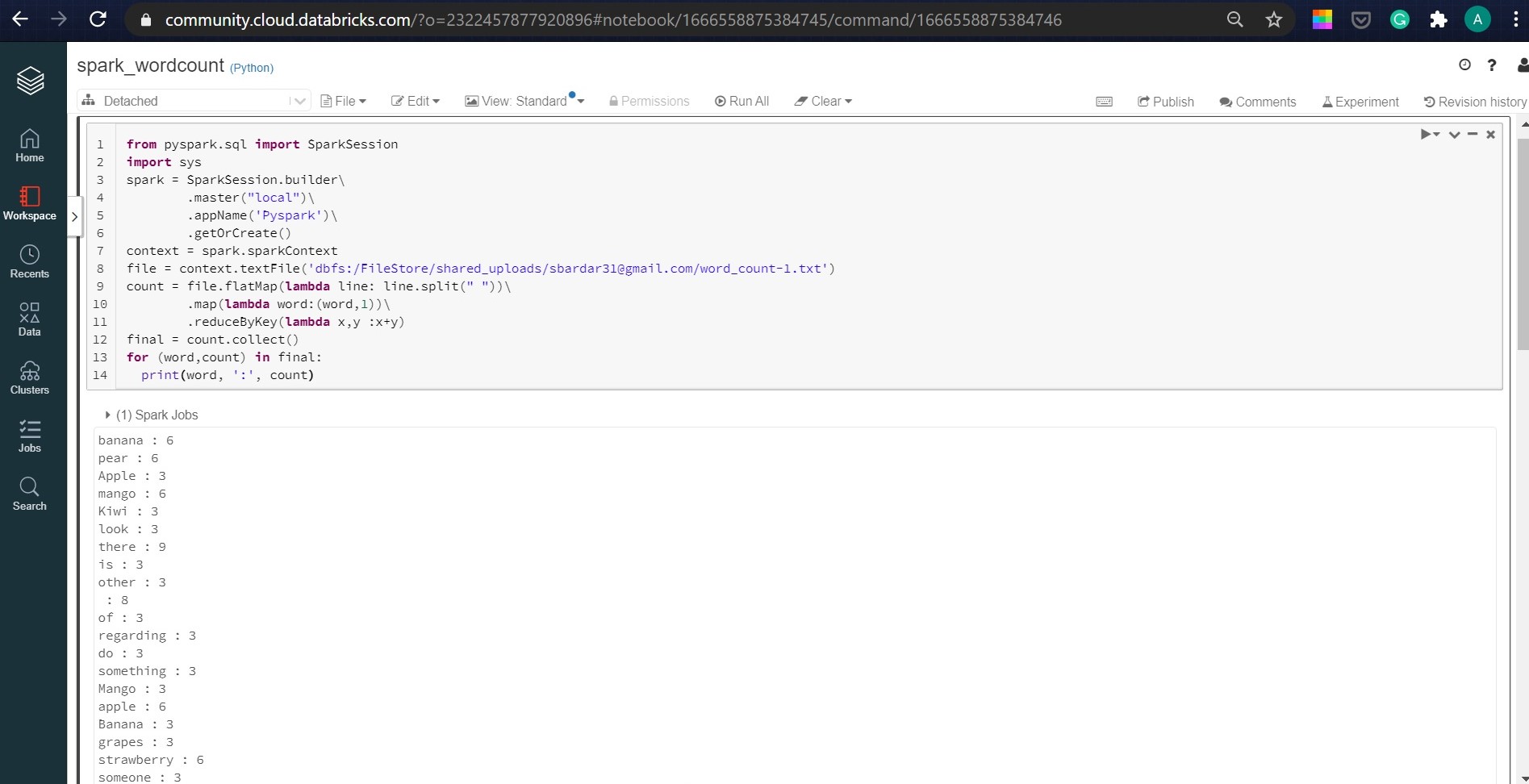
5) Now split the sentence into words and map the occuring words with the value 1.

6) Now to make a unique count on words, applying sum of the values group by words.

7) Now using Collect() method store the values into a variable and print the key and values.

8) End

Code -> Snapshot



**Predefined Functions ->**

* Pyspark Session – It is an entry point for the spark to begin the computation process, it is responsible for all the programically created RDD, DataFrame and Dataset.
* Session.Buider() - Used for creating Session
* GetOrCreate - This method returns an already existing SparkSession, if it does not exists, it creates a new SparkSession.
* Master – Used to define on which the cluster should setup, Example Yarn, Mesos, Standalone. depends on your cluster setup and also uses local[X] when running in Standalone mode. X should be an integer value and should be greater than 0 which represents how many partitions it should create when using RDD, DataFrame, and Dataset. Ideally, the value X should be the number of CPU cores.
* AppName – Used to define the name of the Application.

**Understanding the Internals of it ->**

* How Partitioning is done Internally – Spark is all about RDD, Rdd is resilient distributed data, which is distributed along different nodes in different partitions to achieve parallelism. The spark decides by default the no of partitions, though you can change the no of partition by – ‘val rdd= sc.textFile (“file.txt”, 5)’. To get the no of partition present in rdd you’ll have to do – ‘print(file.getNumPartitions())’. Spark decides the partitioning through ‘**one partition is created for each block of the file in HDFS which is of size 128MB in Hadoop 2.0**’. However, when creating a RDD a second argument can be passed that defines the number of partitions to be created for an RDD.
* How the partition works along with core –> The Partitioning is done so that the many executers take different partition at a time and parallelly execute task given by driver. Now suppose there is 4 cores (executers) to do a task, but the partitions are 5 and each executer takes 5 mins to run. So in total the program takes 10 mins to run and give output cause at a time only 4 executers can run, **hence it Is advised to not make more partitions than no of cores present.**
* How lambda function is applied to each partitioner - >

1. **count = file.flatMap(lambda line: line.split(" "))\** - In this the flat file function is used, as each RDD can be transformed into 0 or More RDD as an output, Here this function is used to take One sentence (RDD) as an input and then splitting into words (one or more RDD). Since the no of Partitions are 2, so at a time 2 lines of sentence will get computed at a time and transform the RDD into a new one.
2. **.map(lambda word:(word,1))\ -** In this map function is used as we want every RDD which is transformed from the above function gets into input and I get one output which is mapped with the given value ‘1’. Hence Every word repeated or non-repeated will get mapped to 1 value. Because of 2 RDD present, 2 words at a time will get mapped with Value ‘1’.
3. **.reduceByKey(lambda x,y :x+y) ->** In this reduceByKey function, every RDD which gets transformed into single RDD gets ingested into this function the value of the repeated key gets summed up so that we get a unique keys of words. Therefore there are 2 RDD present, 2 repeated keys gets summed up.