In [1]:
```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.types import *
```

In [2]:
```
spark = SparkSession\
    .builder\
    .appName("word-count")\
    .getOrCreate()
```

In [3]:
```
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://192.168.0.114:4041)

Version

`v3.0.1`

Master

`local[*]`

AppName

`word-count`

3 approaches to word-count problems

Be careful with the choice of `split` char, we got different answers with `\s+` , using single space char gives the same answer among 3 approaches

Dataframe

In [4]:
```
linesDF = spark.read.text("spark.README.md")
```

In [5]:
```
type(linesDF)
```

Out[5]:
```
pyspark.sql.dataframe.DataFrame
```

In [6]:
```
1 linesDF.select("value").show(5, False)
```

```
+------------------------------------------------------------+
|value                                                       |
+------------------------------------------------------------+
|[Databricks Sandbox](https://community.cloud.databricks.com)|
|                                                            |
|2019-07-30                                                  |
|                                                            |
|reinstall Anaconda                                          |
+------------------------------------------------------------+
only showing top 5 rows
```

In [7]:
```
1 wordCounts = (
2     linesDF
3     .select(F.explode(F.split(F.col("value"), " ")).alias("word"))
4     .groupBy("word").count()
5     .orderBy(F.desc("count"))
6 )
```

In [8]:
```
1 wordCounts.show(10, truncate=False)
```

```
+-------+-----+
|word   |count|
+-------+-----+
|       |85   |
|$      |21   |
|>>>    |14   |
|```    |8    |
|pyspark|8    |
|#      |7    |
|python |7    |
|Spark  |6    |
|##     |6    |
|in     |6    |
+-------+-----+
only showing top 10 rows
```

In [9]:
```
1 wordCounts.count()
```

Out[9]: 235

In [10]:
```
1 wordCounts.rdd.getNumPartitions()
```

Out[10]: 11

In [11]:
```
1 # repartition to 2
2 wordCounts.coalesce(2).write.format("csv").mode("overwrite").save("/
```

filter

In [12]:
```
linesWithSpark = linesDF.filter(F.lower(F.col("value")).contains("sp
linesWithSpark.show(5, False)
```

```
+--------------------------------------------------+
|value                                             |
+--------------------------------------------------+
|Apache Spark                                      |
|http://spark.apache.org/                          |
|$ mkdir -p ~/spark                                |
|$ cd spark                                        |
|$ tar zxvf ~/Downloads/spark-2.4.3-bin-hadoop2.7.tgz|
+--------------------------------------------------+
only showing top 5 rows
```

spark SQL

In [13]:
```
# create a table named "linesTAB"
linesDF.createOrReplaceTempView("linesTAB")
```

In [14]:
```
spark.sql("select * from linesTAB limit 5").show(truncate=False)
```

```
+------------------------------------------------------------+
|value                                                       |
+------------------------------------------------------------+
|[Databricks Sandbox](https://community.cloud.databricks.com)|
|                                                            |
|2019-07-30                                                  |
|                                                            |
|reinstall Anaconda                                          |
+------------------------------------------------------------+
```

In [15]:

```
sql_stmt = """
    with words as (
        select
            explode(split(value, " ")) as word
        from linesTAB
    )
    select
        word,
        count(*) as count
    from words
    group by word
    order by count desc
    limit 10
"""

spark.sql(sql_stmt).show()
```

```
+-------+-----+
|   word|count|
+-------+-----+
|       |   85|
|      $|   21|
|    >>>|   14|
|pyspark|    8|
|    ```|    8|
|      #|    7|
| python|    7|
|     in|    6|
|   with|    6|
|     ##|    6|
+-------+-----+
```

filter line with `spark` word

In [16]:
```
sql_stmt = """
    select
        value
    from linesTAB
    where
    --lower(value) like '%spark%'
    instr(lower(value),'spark') > 0
"""

spark.sql(sql_stmt).show(5, False)
```

```
+--------------------------------------------------+
|value                                             |
+--------------------------------------------------+
|Apache Spark                                      |
|http://spark.apache.org/                          |
|$ mkdir -p ~/spark                                |
|$ cd spark                                        |
|$ tar zxvf ~/Downloads/spark-2.4.3-bin-hadoop2.7.tgz|
+--------------------------------------------------+
only showing top 5 rows
```

RDD

In [17]:
```
sc = spark.sparkContext
```

In [18]:
```
linesRDD = sc.textFile("spark.README.md")
```

In [19]:
```
type(linesRDD)
```

Out[19]:
```
pyspark.rdd.RDD
```

In [20]:
```
linesRDD.take(5)
```

Out[20]:
```
['[Databricks Sandbox](https://community.cloud.databricks.com)',
 '',
 '2019-07-30',
 '',
 'reinstall Anaconda']
```

In [21]:
```
wc = (
    linesRDD.flatMap(lambda x: x.split(" "))
    .map(lambda x: (x, 1))
    .reduceByKey(lambda a,b: a+b)
)
```

In [22]:
```
1 wc.take(5)
```

Out[22]:
```
[('[Databricks', 1),
 ('Sandbox](https://community.cloud.databricks.com)', 1),
 ('', 85),
 ('2019-07-30', 1),
 ('https://www.digitalocean.com/community/tutorials/how-to-install-ana
conda-on-ubuntu-18-04-quickstart',
  2)]
```

In [23]:
```
1 sorted(wc.collect(), key = lambda x: x[1], reverse=True)[:10]
```

Out[23]:
```
[('', 85),
 ('$', 21),
 ('>>>', 14),
 ('```', 8),
 ('pyspark', 8),
 ('python', 7),
 ('#', 7),
 ('Spark', 6),
 ('in', 6),
 ('##', 6)]
```

rdd.save() does not support `overwrite` mode (see https://community.cloudera.com/t5/Support-Questions/Apache-SPARK-Overwrite-data-file/m-p/105253 (https://community.cloudera.com/t5/Support-Questions/Apache-SPARK-Overwrite-data-file/m-p/105253)), one must remove it before hand.

In [24]:
```
1 !rm -rf /tmp/wc_rdd.txt
```

In [25]:
```
1 wc.saveAsTextFile("/tmp/wc_rdd.txt")
```

In [26]:
```
1 !ls /tmp/wc_*
```

```
/tmp/wc_df.csv:
part-00000-48b069f8-66b5-47c4-adcf-5211911a46d0-c000.csv  _SUCCESS
part-00001-48b069f8-66b5-47c4-adcf-5211911a46d0-c000.csv

/tmp/wc_rdd.txt:
part-00000  part-00001  _SUCCESS
```

In [27]:
```
1 spark.stop()
```

In [ ]:
```
1
```