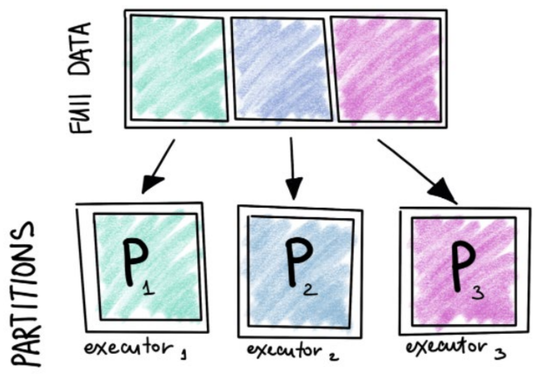
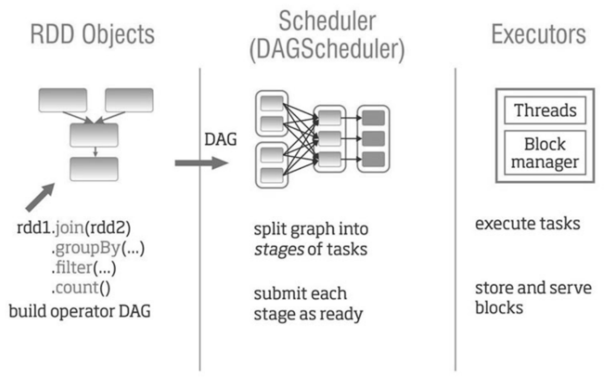
1. **Relations between executors and partitions**

   Ans: As we know that spark engine does parallel processing on a cluster.
   It will distribute the work across the cluster, divide the task into smaller parts, and reduce memory requirements for each node. **Partition** is the main unit of parallelism in Apache Spark. And one the user has submitted his job into the cluster for further processing each partition is sent to a specific **executor**. And it should be noted that only one partition is processed by one executor at a time, so the size and number of partitions transferred to the executor are directly proportional to the time it takes to complete.



2. **RDD and memory management of a spark processing flow.**

   In Spark, there are supported two memory management modes: Static Memory Manager and Unified Memory Manager.
   Spark provides a unified interface Memory Manager for the management of Storage memory and Execution memory. The tasks in the same Executor call the interface to apply for or release memory.
   When an Apache Spark job is run then a DAG (Directed Acyclic Graph) of operators is created, then it is split into a set of Stages, which are further divided into a set of Tasks as shown in the below image. Apache Spark job is split into stages based on the data shuffling (moving data) across machines. Shuffling the data is a costly affair and the less the data shuffled, the faster the job will be completed as shown in the picture



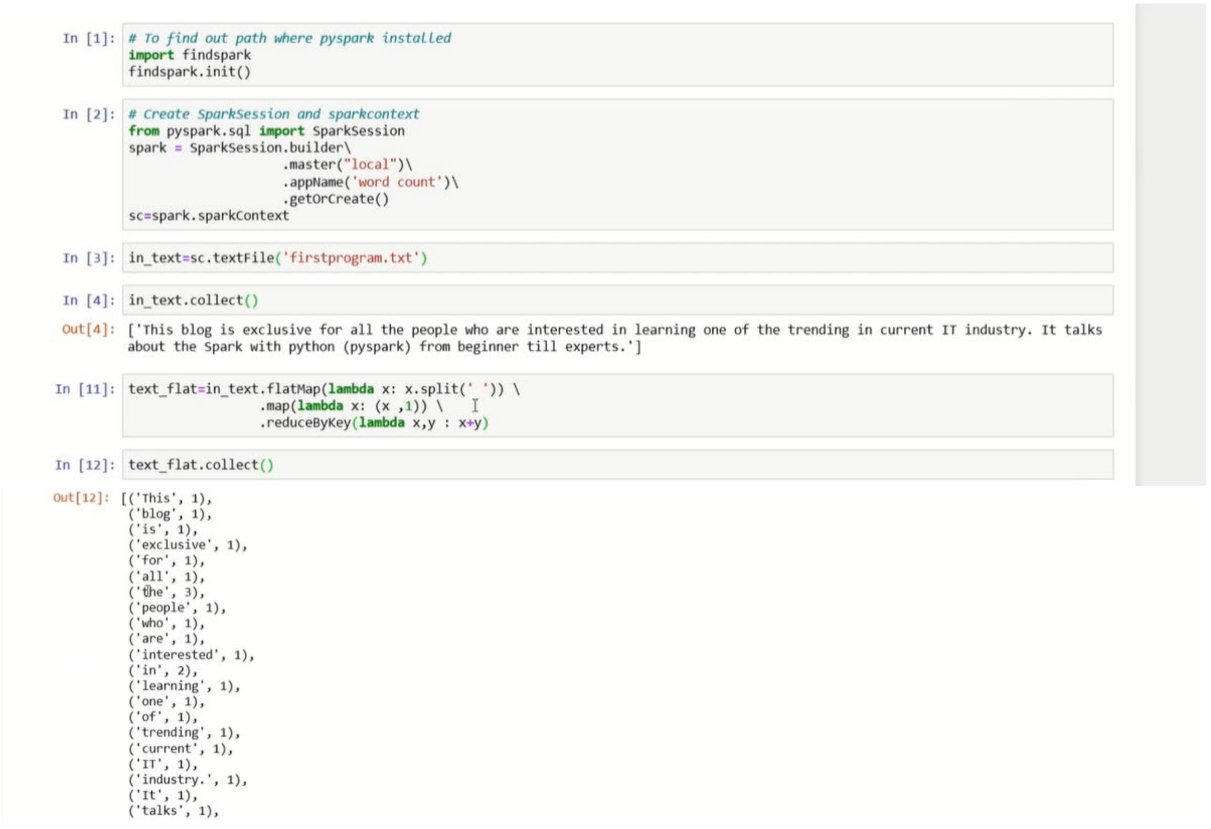
3. Write one spark word count program and highlight relations between executors, partitions, rdd and memory management of a spark processing flow.

step 1 - convert a sentence into words separately
step 2 - map each word with the number 1
step 3 - group each word and add the number

code -

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder\
                ..master("local")\
                .appName('word count')\
                .getorcreate()
sc-spark.sparkContext
in_text=sc.textFile('firstprogram.txt')
in_text.collect()
text_flat=in_text.flatMap(lambda x: x.split(' ')) \
          .map(lambda x: (x,1)) \
          .reduceByKey(lambda x,y: x+y)
text_flat.collect()
```

Output –



```
In [1]:  # To find out path where pyspark installed
         import findspark
         findspark.init()

In [2]:  # Create SparkSession and sparkcontext
         from pyspark.sql import SparkSession
         spark = SparkSession.builder\
                             .master("local")\
                             .appName('word count')\
                             .getOrCreate()
         sc=spark.sparkContext

In [3]:  in_text=sc.textFile('firstprogram.txt')

In [4]:  in_text.collect()

Out[4]:  ['This blog is exclusive for all the people who are interested in learning one of the trending in current IT industry. It talks
         about the Spark with python (pyspark) from beginner till experts.']

In [11]: text_flat=in_text.flatMap(lambda x: x.split(' ')) \
                           .map(lambda x: (x ,1)) \
                           .reduceByKey(lambda x,y : x+y)

In [12]: text_flat.collect()

Out[12]: [('This', 1),
          ('blog', 1),
          ('is', 1),
          ('exclusive', 1),
          ('for', 1),
          ('all', 1),
          ('the', 3),
          ('people', 1),
          ('who', 1),
          ('are', 1),
          ('interested', 1),
          ('in', 2),
          ('learning', 1),
          ('one', 1),
          ('of', 1),
          ('trending', 1),
          ('current', 1),
          ('IT', 1),
          ('industry.', 1),
          ('It', 1),
          ('talks', 1),
```