Import the required libraries then Create SparkContext

In [3]:

```
import findspark
findspark.init()

import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext
```

```
21/10/22 14:37:22 WARN Utils: Your hostname, yousri-Lenovo-Legion-5-15
IMH05H resolves to a loopback address: 127.0.1.1; using 192.168.1.105
instead (on interface wlp0s20f3)
21/10/22 14:37:22 WARN Utils: Set SPARK_LOCAL_IP if you need to bind t
o another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/opt/spark/jars/spark-unsafe_2.12-3.0.1.jar) to constructor jav
a.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apac
he.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illeg
al reflective access operations
WARNING: All illegal access operations will be denied in a future rele
ase
21/10/22 14:37:33 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicab
le
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.p
roperties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
```

Create and display an RDD from the following list

In [4]:

```
List = [('JK', 22), ('V', 24), ('Jimin',24), ('RM', 25), ('J-Hope', 25), ('Suga', 2
```

In [9]:

```
listRDD = sc.parallelize(List)
listRDD.collect()
```

Out[9]:

```
[('JK', 22),
 ('V', 24),
 ('Jimin', 24),
 ('RM', 25),
 ('J-Hope', 25),
 ('Suga', 26),
 ('Jin', 27)]
```

Read sample1.txt file into RDD and displaying the first 4 elements

In [11]:

```
distFile = sc.textFile('sample1.txt')
distFile
```

Out[11]:

```
sample1.txt MapPartitionsRDD[8] at textFile at NativeMethodAccessorImp
l.java:0
```

Count the total number of rows in RDD

In [12]:

```
distFile.count()
```

Out[12]:

```
7
```

Create a function to convert the data into lower case and splitting it

In [14]:

```
lstofRows = distFile.collect()
lstofRows
```

Out[14]:

```
['Utilitatis causa amicitia est quaesita.',
 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. ',
 'Collatio igitur ista te nihil iuvat. Honesta oratio, Socratica, Plat
onis etiam. Primum in nostrane potestate est, quid meminerimus? ',
 'Duo Reges: constructio interrete. ',
 'Quid, si etiam iucunda memoria est praeteritorum malorum? Si quidem,
inquit, tollerem, sed relinquo. An nisi populari fama?',
 '',
 'Quamquam id quidem licebit iis existimare, qui legerint. Summum a vo
bis bonum voluptas dicitur. At hoc in eo M. Refert tamen, quo modo. Qu
id sequatur, quid repugnet, vident. Iam id ipsum absurdum, maximum mal
um neglegi.']
```

In [18]:

```
"""
    dataToLowerSplit - Function
    Used to convert textfile line by line into lower case then split it into words
"""
def dataToLowerSplit(distFile):
    return distFile.map(lambda line: line.lower().split()).collect()

dataToLowerSplit(distFile)
```

Out[18]:

```
[['utilitatis', 'causa', 'amicitia', 'est', 'quaesita.'],
 ['lorem',
  'ipsum',
  'dolor',
  'sit',
  'amet,',
  'consectetur',
  'adipiscing',
  'elit.'],
 ['collatio',
  'igitur',
  'ista',
  'te',
  'nihil',
  'iuvat.',
  'honesta',
  'oratio,',
  'socratica,',
  'platonis',
  'etiam.',
  'primum',
  'in',
  'nostrane',
  'potestate',
  'est,',
  'quid',
  'meminerimus?'],
 ['duo', 'reges:', 'constructio', 'interrete.'],
 ['quid,',
  'si',
  'etiam',
  'iucunda',
  'memoria',
  'est',
  'praeteritorum',
  'malorum?',
  'si',
  'quidem,',
  'inquit,',
  'tollerem,',
  'sed',
  'relinquo.',
  'an',
  'nisi',
  'populari',
  'fama?'],
 [],
```

```
['quamquam',
 'id',
 'quidem',
 'licebit',
 'iis',
 'existimare,',
 'qui',
 'legerint.',
 'summum',
 'a',
 'vobis',
 'bonum',
 'voluptas',
 'dicitur.',
 'at',
 'hoc',
 'in',
 'eo',
 'm.',
 'refert',
 'tamen,',
 'quo',
 'modo.',
 'quid',
 'sequatur,',
 'quid',
 'repugnet,',
 'vident.',
 'iam',
 'id',
 'ipsum',
 'absurdum,',
 'maximum',
 'malum',
 'neglegi.']]
```

Filter the stopwords from the previous text

In [22]:

```
stopwords = ['a','all','the','as','is','am','an','and',
             'be','been','from','had','I','I'd','why','with']
# Hint: you may need use flatMap
```

In [33]:

```
distFile.flatMap(lambda line: line.lower().split()).filter(lambda word: word not in
```

Out[33]:

```
['utilitatis',
 'causa',
 'amicitia',
 'est',
 'quaesita.',
 'lorem',
 'ipsum',
 'dolor',
 'sit',
 'amet,',
 'consectetur',
 'adipiscing',
 'elit.',
 'collatio',
 'igitur',
 'ista',
 'te',
 'nihil',
 'iuvat.',
 'honesta',
 'oratio,',
 'socratica,',
 'platonis',
 'etiam.',
 'primum',
 'in',
 'nostrane',
 'potestate',
 'est,',
 'quid',
 'meminerimus?',
 'duo',
 'reges:',
 'constructio',
 'interrete.',
 'quid,',
 'si',
 'etiam',
 'iucunda',
 'memoria',
 'est',
 'praeteritorum',
 'malorum?',
 'si',
 'quidem,',
 'inquit,',
 'tollerem,',
 'sed',
 'relinquo.',
 'nisi',
 'populari',
 'fama?',
 'quamquam',
 'id',
```

```
 'quidem',
 'licebit',
 'iis',
 'existimare,',
 'qui',
 'legerint.',
 'summum',
 'vobis',
 'bonum',
 'voluptas',
 'dicitur.',
 'at',
 'hoc',
 'in',
 'eo',
 'm.',
 'refert',
 'tamen,',
 'quo',
 'modo.',
 'quid',
 'sequatur,',
 'quid',
 'repugnet,',
 'vident.',
 'iam',
 'id',
 'ipsum',
 'absurdum,',
 'maximum',
 'malum',
 'neglegi.']
```

In [34]:

```
latinStopwords = ['a', 'omnis', 'omnis', 'as', 'is', 'am', 'an', 'et',
             'esse', 'fuisse', 'ex', 'habui', 'I', 'fui', 'cur', 'cum']
```

In [35]:

```
distFile.flatMap(lambda line: line.lower().split()).filter(lambda word: word not in
```

Out[35]:

```
['utilitatis',
 'causa',
 'amicitia',
 'est',
 'quaesita.',
 'lorem',
 'ipsum',
 'dolor',
 'sit',
 'amet,',
 'consectetur',
 'adipiscing',
 'elit.',
 'collatio',
 'igitur',
 'ista',
 'te',
 'nihil',
 'iuvat.',
 'honesta',
 'oratio,',
 'socratica,',
 'platonis',
 'etiam.',
 'primum',
 'in',
 'nostrane',
 'potestate',
 'est,',
 'quid',
 'meminerimus?',
 'duo',
 'reges:',
 'constructio',
 'interrete.',
 'quid,',
 'si',
 'etiam',
 'iucunda',
 'memoria',
 'est',
 'praeteritorum',
 'malorum?',
 'si',
 'quidem,',
 'inquit,',
 'tollerem,',
 'sed',
 'relinquo.',
 'nisi',
 'populari',
 'fama?',
 'quamquam',
 'id',
```

```
 'quidem',
 'licebit',
 'iis',
 'existimare,',
 'qui',
 'legerint.',
 'summum',
 'vobis',
 'bonum',
 'voluptas',
 'dicitur.',
 'at',
 'hoc',
 'in',
 'eo',
 'm.',
 'refert',
 'tamen,',
 'quo',
 'modo.',
 'quid',
 'sequatur,',
 'quid',
 'repugnet,',
 'vident.',
 'iam',
 'id',
 'ipsum',
 'absurdum,',
 'maximum',
 'malum',
 'neglegi.']
```

In [29]:

```
st are in english not in lattin, and still after translate
into lattin, still not the same output, so this is must be not the complete output.
```

Out[29]:

```
['utilitatis',
 'causa',
 'amicitia',
 'est',
 'quaesita.',
 'lorem',
 'ipsum',
 'dolor',
 'sit',
 'amet,']
```

Filter the words starting with ‘c’

In [42]:

```
ower().split()).filter(lambda word: word if(word.startswith('c')) else 0).collect()
```

Out[42]:

```
['causa', 'consectetur', 'collatio', 'constructio']
```

Reduce the data by key and sum it (use the data from the following list)

In [43]:

```
List = [('JK', 22), ('V', 24), ('Jimin',24), ('RM', 25)
        , ('J-Hope', 25), ('Suga', 26), ('Jin', 27)
       , ('J-Hope', 12), ('Suga', 25), ('Jin', 34)
       , ('JK', 32), ('V', 44), ('Jimin',14), ('RM', 35)]
# Hint: use reduceByKey
```

In [49]:

```
lstRDD = sc.parallelize(List)
print(lstRDD.collect())

from operator import add
lstRDD.reduceByKey(add).collect()
```

```
[('JK', 22), ('V', 24), ('Jimin', 24), ('RM', 25), ('J-Hope', 25), ('S
uga', 26), ('Jin', 27), ('J-Hope', 12), ('Suga', 25), ('Jin', 34), ('J
K', 32), ('V', 44), ('Jimin', 14), ('RM', 35)]
```

Out[49]:

```
[('JK', 54),
 ('Jimin', 38),
 ('RM', 60),
 ('Jin', 61),
 ('V', 68),
 ('J-Hope', 37),
 ('Suga', 51)]
```

In [25]:

```
#required Output
```

Out[25]:

```
[('Suga', 51),
 ('Jin', 61),
 ('JK', 54),
 ('V', 68),
 ('Jimin', 38),
 ('RM', 60),
 ('J-Hope', 37)]
```

Creat some key value pairs RDDs

In [50]:

```
rdd1 = sc.parallelize([('a',2),('b',3)])
rdd2 = sc.parallelize([('a',9),('b',7),('c',10)])
```

Perform Join operation on the RDDs (rdd1,rdd2)

In [51]:

```
rdd1.join(rdd2).collect()
```

Out[51]:

```
[('b', (3, 7)), ('a', (2, 9))]
```

In [33]:

```
#required OUTPUT
```

Out[33]:

```
[('b', (3, 7)), ('a', (2, 9))]
```

In [ ]:

```

```