Import the required libraries then Create SparkContext

In [1]:
```
import pyspark
import findspark

findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
from pyspark import SparkContext
sc = SparkContext()
spark = SparkSession.builder.appName("Assignment 1").getOrCreate()
sc = spark.sparkContext
```

Create and display an RDD from the following list

In [3]:
```
lst = [('JK', 22), ('V', 24), ('Jimin',24), ('RM', 25), ('J-Hope', 25), ('Sug
a', 26), ('Jin', 27)]
```

In [4]:
```
rdd1 = sc.parallelize(lst)
rdd1.collect()
```

Out[4]:
```
[('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 [5]:
```
text = sc.textFile('sample1.txt')
text.take(2)
```

Out[5]:
```
['Utilitatis causa amicitia est quaesita.',
 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. ']
```

Count the total number of rows in RDD

In [6]:
```
text.count()
```

Out[6]:
```
7
```

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

In [7]:
```
text_lower = text.map(lambda x: x.lower())
test_split = text_lower.flatMap(lambda line: line.split())
test_split.collect()
```

```
Out[7]: ['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.']
```

In [ ]:

```
```

Filter the stopwords from the previous text

In [8]:

```
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 [9]:
```
def func(x):
    if x not in stopwords:
        return x

text_stopwords_filtered =  test_split.map(func)
text_stopwords_filtered.collect()
```

```
Out[9]: ['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.',
 None,
 'nisi',
 'populari',
 'fama?',
 'quamquam',
 'id',
 'quidem',
 'licebit',
 'iis',
 'existimare,',
 'qui',
 'legerint.',
 'summum',
 None,
 '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 [ ]:

Filter the words starting with ‘c’

In [ ]:

```
def func2(x):
    print("---")
    if x[0] == 'c':
        return x

te= text_stopwords_filtered.map(func2)
te.collect()
```

In [ ]:

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

```
In [ ]: lst = [('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 [ ]: rdd = sc.parallelize(lst)
        rdd.reduceByKey(sum).collect()
```

```
In [ ]:
```

Creat some key value pairs RDDs

```
In [ ]: 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 [33]:
Out[33]: [('b', (3, 7)), ('a', (2, 9))]
```

```
In [ ]:
```