ICP2_Siva_BigdataAnalytics.py

```
from pyspark.conf import SparkConf
from pyspark.context import SparkContext

#Creating Basic RDD (pyspark.rdd.RDD) using SparkConf with the help of master
spark_conf = SparkConf().setAppName("BigData Siva ICP2").setMaster("local[*]")
spark_context = SparkContext(conf=spark_conf)

content_rdd_text = spark_context.textFile("icp2.txt")3,860.40
print(type(content_rdd_text))
nonempty_lines = content_rdd_text.filter(lambda x: len(x) > 0)
#  Similar to map, it returns a new RDD by applying  a function to each element of the RDD, but output is flattened.
lower_words = nonempty_lines.flatMap(lambda x: x.lower().split(" "))
print(lower_words.count())
# It returns a new RDD by applying a function to each element of the RDD.   Function in map can return only one item.
result_groupby = lower_words.map(lambda x: (x[0], x)).groupByKey().map(lambda x: (x[0], list(x[1])))
print("--------------------Using groupby----------------------------------------------------")
# Retreive the elements one by one using foreach loop
for each_groupby in result_groupby.collect():
   print(each_groupby)
result_reducedby = lower_words.map(lambda x: (x[0], x)).reduceByKey(lambda x, y: x + " , " + y)
print("--------------------------------------------------------------------------------------")
print(lower_words.count())
print("--------------------Using reducedby--------------------------------------------------")
# Retreive the elements one by one using foreach loop
for each_reducedby in result_reducedby.collect():
   print(each_reducedby)
```

ICP2 Siva Big Data Analytics Unique Words.py

```
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import SQLContext
from pyspark.sql.functions import regexp_replace, trim, col, lower
from pyspark.sql.functions import desc
from pyspark.sql.functions import split, explode
from pyspark.sql.functions import countDistinct, avg, stddev
import re
import pyspark.sql.functions as f

# Create the SparkContent
sc = pyspark.SparkContext()
sqlContext = SQLContext(sc)

# remove puncutation marks in the given text
def removePunctuation(column):
   return trim(lower(regexp_replace(column, '([^\s\w_a-zA-Z\[0-9]]|_)+', ''))).alias('sentence')
   #return trim(lower(regexp_replace(column, '([^\s\w_]|_)+', ''))).alias('sentence')


# no.of.words words in the given text
def wordCount(wordListDF):
   return wordListDF.groupBy('word').count()

# Display the type of the Spark sqlContext
print(type(sqlContext))

fileName = "icp2.txt"

#read the input file and remove puncutations if any
DF_New = sqlContext.read.text(fileName).select(removePunctuation(col('value')))
DF_New.show(truncate=False)

# split the input text and keep column name as sentence
shakeWordsSplitDF = (DF_New.select(split(DF_New.sentence, '\s+').alias('split')))
# provide alias name as word by replacing sentence
shakeWordsSingleDF = (shakeWordsSplitDF.select(explode(shakeWordsSplitDF.split).alias('word')))
# retreive only non empty words presented in the text
```

```
shakeWordsDF = shakeWordsSingleDF.where(shakeWordsSingleDF.word != '')
# dimension of the data frame
print("shape of the data: ({},{})".format(shakeWordsDF.count(),len(shakeWordsDF.dtypes)))


# Count the words
from pyspark.sql.functions import desc
WordsAndCountsDF = wordCount(shakeWordsDF)
# displayting data frame in the descending order
topWordsAndCountsDF = WordsAndCountsDF.orderBy("word", ascending=False)
topWordsAndCountsDF.show(n=108)
```