In [1]:
```
#imports
import pyspark
from pyspark.sql import SparkSession
from pyspark.conf import SparkConf
import pyspark.sql.functions as F
from pyspark.sql.functions import col
from pyspark.sql.functions import monotonically_increasing_id
from pyspark.sql.functions import row_number
from pyspark.sql.window import Window
import pandas as pd
import math

#Initializing Spark Conf
conf=SparkConf()\
        .setMaster("local[*]")\
        .setAppName("WordCount")\
        .setExecutorEnv("spark.executor.memory","1g")\
        .setExecutorEnv("spark.driver.memory","1g")

#Creating Spark Session
spark=SparkSession.builder\
        .config(conf=conf)\
        .getOrCreate()
```

In [2]:
```
#Spark context
sc=spark.sparkContext
```

In [10]:
```
#text file path
textfile="/opt/awsAssignment/sample-a.txt"
out_text= "/opt/awsAssignment/sample-a-out.txt"
out_file_header = "              Output for Sample - a                    "
```

In [11]:
```
#Importing textfile as rdd
word_rdd=sc.textFile(textfile)
```

In [12]:
```
#Function to remove punc and lowercase
def lower_clean_str(x):
  punc='!"#$%&\'()*+,./:;<=>?@[\\]^_`{|}~1 2 3 4 5 6 7 8 9 0 -'
  lowercased_str = x.lower()
  for ch in punc:
    lowercased_str = lowercased_str.replace(ch, ' ')
  return lowercased_str
```

In [13]:
```
#Filtered RDD
filtered_rdd = word_rdd.map(lower_clean_str)
```

In [14]:
```
#Separate Words By " "
separatedword_rdd=filtered_rdd.flatMap(lambda word: word.split(" "))
```

In [15]:
```
#Removing white spaces and empty fields
separatedword_rdd = separatedword_rdd.filter(lambda x:x!='')
```

In [16]:
```
#Adding values to each word
word_with_value=separatedword_rdd.map(lambda  word:(word,1))
total_words = word_with_value.count()
total_words
```

Out[16]: 53

In [20]:
```
#Reduces by key(word)
word_with_value_red=word_with_value.reduceByKey(lambda x,y:(x+y)).sortByKey()
distinct_word_count = word_with_value_red.count()
```

In [21]:
```
#Changeing key and value positions
word_count=word_with_value_red.map(lambda x:(x[1],x[0]))
```

In [22]:
```
#Sort by Frequency
wc_sort = word_count.sortByKey(False).collect()
```

In [23]:
```
#Creating a Dataframe using above RDD
word_count_rdd = spark.sparkContext.parallelize(wc_sort)
columns = ["Frequency","Word"]
word_count_df = word_count_rdd.toDF(columns)
```

In [24]:
```
#Adding Rank column
wc = word_count_df.withColumn("rank",row_number().over(Window.orderBy(monotonically_inc
```

In [25]:
```
wc.show(100)
```

```
+---------+-----------+----+
|Frequency|       Word|rank|
+---------+-----------+----+
|        3|      spark|   1|
|        3|        the|   2|
|        2|         an|   3|
|        2|     apache|   4|
|        1|     amplab|   5|
|        1|        and|   6|
|        1|         at|   7|
|        1|   berkeley|   8|
|        1| california|   9|
|        1|    cluster|  10|
|        1|   clusters|  11|
|        1|   codebase|  12|
|        1|  computing|  13|
|        1|       data|  14|
|        1|  developed|  15|
|        1|distributed|  16|
|        1|    donated|  17|
|        1|     entire|  18|
|        1|      fault|  19|
|        1|        for|  20|
|        1| foundation|  21|
|        1|  framework|  22|
|        1|    general|  23|
|        1|        has|  24|
|        1|   implicit|  25|
|        1|  interface|  26|
|        1|         is|  27|
|        1|         it|  28|
|        1|      later|  29|
|        1| maintained|  30|
|        1|         of|  31|
|        1|       open|  32|
|        1| originally|  33|
|        1|parallelism|  34|
|        1|programming|  35|
|        1|   provides|  36|
```

```
|        1|    purpose|  37|
|        1|          s|  38|
|        1|      since|  39|
|        1|   software|  40|
|        1|     source|  41|
|        1|         to|  42|
|        1|  tolerance|  43|
|        1| university|  44|
|        1|        was|  45|
|        1|      which|  46|
|        1|       with|  47|
+---------+-----------+----+
```

In [26]:
```
#Values in Datafeame
wc_val = wc.count()
```

In [27]:
```
#Calculating Popular words
print("Popular words")
import math


popthreshold = math.ceil(wc_val * 5 /100)
print(popthreshold)

popularwords = wc.select('rank','Word','Frequency').filter(wc.rank <= popthreshold)
popularwords.show()
popularwordspd = popularwords.toPandas()
```

```
Popular words
3
+----+-----+---------+
|rank| Word|Frequency|
+----+-----+---------+
|   1|spark|        3|
|   2|  the|        3|
|   3|   an|        2|
+----+-----+---------+
```

In [28]:
```
#Calculating Common words
print("Common words")

lowerthreshold = math.floor(wc_val * 47.5 /100)
upperthreshold = math.ceil(wc_val * 52.5 /100)
print(lowerthreshold)
print(upperthreshold)

commonwords = wc.select('rank','Word','Frequency').filter((wc.rank >=  lowerthreshold)

commonwords.show()
commonwordspd = commonwords.toPandas()
```

```
Common words
22
25
+----+---------+---------+
|rank|     Word|Frequency|
+----+---------+---------+
|  22|framework|        1|
|  23|  general|        1|
|  24|      has|        1|
|  25| implicit|        1|
```

```
+----+---------+---------+
```

In [29]:
```
#Calculating Rare words
print("Rare words")


rarethreshold = math.floor(wc_val * 95 /100)
print(rarethreshold)

rarewords = wc.select('rank','Word','Frequency').filter(wc.rank >= rarethreshold)

rarewords.show()
rarewordspd = rarewords.toPandas()
```

```
Rare words
44
+----+----------+---------+
|rank|      Word|Frequency|
+----+----------+---------+
|  44|university|        1|
|  45|       was|        1|
|  46|     which|        1|
|  47|      with|        1|
+----+----------+---------+
```

In [31]:
```
# Letters
char_counts_with_value_red = word_with_value.flatMap(lambda each: each[0]).map(lambda c
char_counts_with_value_red.count()
```

Out[31]:
```
317
```

In [32]:
```
#Character count reduced by char
char_counts_with_value_red = word_with_value.flatMap(lambda each: each[0]).map(lambda c
    .map(lambda c: (c, 1)).reduceByKey(lambda v1, v2: v1 + v2)
```

In [33]:
```
#Changeing key value position
char_count=char_counts_with_value_red.map(lambda x:(x[1],x[0]))
```

In [34]:
```
#Sort by frequency
cc_sort = char_count.sortByKey(False).collect()
```

In [35]:
```
#Creating DF using RDD
char_count_rdd = spark.sparkContext.parallelize(cc_sort)
columns = ["Frequency","Letter"]
char_count_df = char_count_rdd.toDF(columns)
```

In [36]:
```
#Adding ranking column
cc = char_count_df.withColumn("Rank",row_number().over(Window.orderBy(monotonically_inc
cc.show(26)
```

```
+---------+------+----+
|Frequency|Letter|Rank|
+---------+------+----+
|       36|     e|   1|
|       35|     a|   2|
|       25|     r|   3|
|       25|     i|   4|
|       24|     t|   5|
|       19|     s|   6|
```

```
|       19|     n|   7|
|       19|     o|   8|
|       16|     l|   9|
|       15|     p|  10|
|       13|     c|  11|
|       12|     d|  12|
|        9|     h|  13|
|        9|     u|  14|
|        8|     m|  15|
|        8|     f|  16|
|        5|     g|  17|
|        5|     k|  18|
|        5|     w|  19|
|        4|     b|  20|
|        3|     y|  21|
|        3|     v|  22|
+---------+------+----+
```

In [37]:
```
#Dataframe Size
cc_val = cc.count()
```

In [38]:
```
#Calculating Popular Letters
print("Popular Letters")


popthresholdcc = math.ceil(cc_val * 5 /100)
print(popthresholdcc)

popularchars = cc.select('Rank','Letter','Frequency').filter(cc.Rank <= popthresholdcc)
popularchars.show()
popularcharspd = popularchars.toPandas()
```

```
Popular Letters
2
+----+------+---------+
|Rank|Letter|Frequency|
+----+------+---------+
|   1|     e|       36|
|   2|     a|       35|
+----+------+---------+
```

In [39]:
```
#Calculating Common Letters
print("Common Letters")

lowerthresholdcc = math.floor(cc_val * 47.5 /100)
upperthresholdcc = math.ceil(cc_val * 52.5 /100)
print(lowerthresholdcc)
print(upperthresholdcc)

commonchars = cc.select('Rank','Letter','Frequency').filter((cc.Rank >=  lowerthreshold
commonchars.show()
commoncharspd = commonchars.toPandas()
```

```
Common Letters
10
12
+----+------+---------+
|Rank|Letter|Frequency|
+----+------+---------+
|  10|     p|       15|
|  11|     c|       13|
```

```
|  12|     d|       12|
+----+------+---------+
```

In [40]:
```
#Calculating Rare Letters
print("Rare words")


rarethresholdcc = math.floor(cc_val * 95 /100)
print(rarethresholdcc)

rareletters = cc.select('Rank','Letter','Frequency').filter(cc.Rank >= rarethresholdcc)
rateletterspd = rareletters.toPandas()
rareletters.show()
```

```
Rare words
20
+----+------+---------+
|Rank|Letter|Frequency|
+----+------+---------+
|  20|     b|        4|
|  21|     y|        3|
|  22|     v|        3|
+----+------+---------+
```

In [30]:
```
#Printing into output file
```

In [31]:
```
print("total number of words=" + str(total_words))
```

```
total number of words=2186595
```

In [32]:
```
f = open(out_text, "a")
f.write("-----------------------------------------------\n")
f.write(out_file_header+ "\n")
f.write("-----------------------------------------------\n\n")
f.write("total number of words = " + str(total_words)+"\n")
f.write("total number of distinct words = " + str(distinct_word_count)+"\n")
f.write("popular_threshold_word = " + str(popthreshold)+"\n")
f.write("common_threshold_l_word = " + str(lowerthreshold)+"\n")
f.write("common_threshold_u_word = " + str(upperthreshold)+"\n")
f.write("rare_threshold_word = " + str(rarethreshold)+"\n")
f.write("-----------------------------------------------\n\n")

f.write("Popular words \n")
f.write(str(popularwordspd))
f.write("\n\n")

f.write("Common words \n")
f.write(str(commonwordspd))
f.write("\n\n")


f.write("Rare words \n")
f.write(str(rarewordspd))
f.write("\n\n")


f.write("-----------------------------------------------\n\n")
f.write("total number of distinct letters = " + str(cc_val)+"\n")
f.write("popular_threshold_letters = " + str(popthresholdcc)+"\n")
f.write("common_threshold_l_letters = " + str(lowerthresholdcc)+"\n")
```

```
f.write("common_threshold_u_letters = " + str(upperthresholdcc)+"\n")
f.write("rare_threshold_letters = " + str(rarethresholdcc)+"\n")
f.write("-------------------------------------------\n\n")

f.write("Popular Letters \n")
f.write(str(popularcharspd))
f.write("\n\n")

f.write("Common Letters \n")
f.write(str(commoncharspd))
f.write("\n\n")


f.write("Rare Letters \n")
f.write(str(rateletterspd))
f.write("\n\n")

f.close()

print("File Writing Complete")
```

```
File Writing Complete
```

In [ ]: