databricksHW_02

DSCI 417 – Homework 02

Sean Graham

```
from pyspark.sql import SparkSession
import pandas as pd
from string import punctuation


spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext


ws_lines = sc.textFile('/FileStore/tables/shakespeare_complete.txt')

ws_words = (ws_lines.flatMap(lambda x : x.split(' ')).flatMap(lambda x : x.split('-')) .flatMap(lambda x : x.split('_')) .flatMap(lambda x : x.split('.'))
.flatMap(lambda x : x.split(',')) .flatMap(lambda x : x.split(', ')) .flatMap(lambda x : x.split('|')) .flatMap(lambda x : x.split('\t')).map(lambda x :
x.strip(punctuation)).map(lambda x : x.strip('0123456789')).map(lambda x : x.replace("'", '')).map(lambda x : x.lower()).filter(lambda x : x != ''))

dist_words = ws_words.distinct()

print('Total number of words:    ', ws_words.count())
print('Number of distinct words: ', dist_words.count())
```

```
Total number of words:     887278
Number of distinct words:  25363
```

```
sample = ws_words.sample(withReplacement=False, fraction=0.0001)
print(sample.collect())
```

```
['the', 'for', 'ages', 'mine', 'you', 'charmian', 'be', 'under', 'france', 'church', 'his', 'may', 'old', 'friend', 'to', 'rats', 'stand', 'where', 'are', 'kno
w', 'sir', 'my', 'terms', 'and', 'that', 'the', 'am', 'a', 'to', 'pray', 'lock', 'talbot', 'purge', 'master', 'my', 'his', 'must', 'tis', 'you', 'tower', 'to',
'to', 'remote', 'these', 'the', 'parting', 'heart', 'do', 'juvenal', 'the', 'costard', 'berowne', 'fair', 'berowne', 'since', 'their', 'creeps', 'came', 'duke',
'accounted', 'and', 'falstaff', 'vere', 'yet', 'vanity', 'go', 'moral', 'god', 'mistress', 'an', 'jealous', 'in', 'me', 'some', 'life', 'with', 'i', 'boy', 'bia
nca', 'thou', 'bound', 'his', 'you', 'the', 'cassandra', 'hector', 'i', 'of', 'a', 'me', 'intend', 'th', 'to']
```

Problem 2: Longest Words

```
def length(x, y):
    if len(x) < len(y):
        return y
    if len(y) < len(x):
        return x
    if x <= y:
        return y
    return x

longest_word = dist_words.reduce(length)
print(longest_word)
```

```
honorificabilitudinitatibus
```

```
sorted_length = dist_words.sortBy(lambda x : len(x), ascending=False)

for word in sorted_length.take(20):
    print(word)
```

```
honorificabilitudinitatibus
anthropophaginian
```

```
undistinguishable
indistinguishable
incomprehensible
northamptonshire
superserviceable
perpendicularly
notwithstanding
gloucestershire
excommunication
interchangeably
enfranchisement
uncompassionate
distinguishment
portotartarossa
distemperatures
northumberlands
prognostication
unreconciliable
```

Problem 3: Word Frequency

```
pairs = ws_words.map(lambda x : (x, 1))

word_counts = pairs.reduceByKey(lambda x, y : x + y).sortBy(lambda x : x[1], ascending=False)

word_count_list = word_counts.collect()[:20]
df = pd.DataFrame(word_count_list, columns=['Word', 'Count'])

df

Out[7]:
```

	Word	Count
0	the	27379
1	and	26082
2	i	20717
3	to	19661
4	of	17474
5	a	14723
6	you	13630
7	my	12489
8	in	10996
9	that	10915
10	is	9137
11	not	8512
12	with	7778
13	me	7776
14	it	7692
15	for	7578
16	be	6867
17	his	6859
18	your	6658
19	this	6606

Problem 4: Removing Stop Words

```
sw_rdd = sc.textFile('/FileStore/tables/stopwords.txt')

print(sw_rdd.count())

sample2 = sw_rdd.sample(withReplacement=False, fraction=0.05)
print(sample2.collect())

sw = sw_rdd.collect()
```

```
668
['affected', 'becomes', 'does', 'doing', 'forth', 'found', 'hed', 'hers', 'invention', 'know', 'lets', 'must', 'our', 're', 'research', 'same', "she'll", 'speci
fically', 'stop', 'thereto', 'truly', 'two', 'unlikely', 'very', 'want', 'we', 'widely', 'yet', 'your']
```

```
ws_words_f = (ws_words.filter(lambda x : x not in sw))
dist_words_f = ws_words_f.distinct()
print('Number of Distinct Non-Stop Words: ', dist_words_f.count())
```

```
Number of Distinct Non-Stop Words:  24841
```

```
pairs = ws_words_f.map(lambda x : (x, 1))

word_counts = pairs.reduceByKey(lambda x, y : x + y).sortBy(lambda x : x[1], ascending=False)

word_count_list = word_counts.collect()[:20]
df = pd.DataFrame(word_count_list, columns=['Word', 'Count'])

df
```

Out[10]:

	Word	Count
0	will	4977
1	thy	4034
2	thee	3180
3	lord	3062
4	king	2871
5	good	2834
6	sir	2763
7	well	2553
8	enter	2350
9	love	2109
10	ill	2024
11	hath	1942
12	man	1876
13	tis	1408
14	speak	1169
15	mine	1165
16	time	1074
17	duke	1071
18	exeunt	1035
19	heart	1012

Problem 5: Diamonds Dataset

```
diamonds_raw = sc.textFile('/FileStore/tables/diamonds.txt')
print(diamonds_raw.count())
```

```
53941
```

```
for row in diamonds_raw.take(5):
    print(row)
```

```
carat   cut     color   clarity depth   table   price   x       y       z
0.23    Ideal   E       SI2     61.5    55      326     3.95    3.98    2.43
0.21    Premium E       SI1     59.8    61      326     3.89    3.84    2.31
0.23    Good    E       VS1     56.9    65      327     4.05    4.07    2.31
0.29    Premium I       VS2     62.4    58      334     4.2     4.23    2.63
```

```
def process_row(row):
    tokens = row.split('\t')
    return (float(tokens[0]), str(tokens[1]), str(tokens[2]), str(tokens[3]), float(tokens[4]), float(tokens[5]), int(tokens[6]), float(tokens[7]),
float(tokens[8]), float(tokens[9]))

diamonds = (diamonds_raw.filter(lambda x : 'carat' not in x).map(process_row))

for row in diamonds.take(5):
    print(row)
```

```
(0.23, 'Ideal', 'E', 'SI2', 61.5, 55.0, 326, 3.95, 3.98, 2.43)
(0.21, 'Premium', 'E', 'SI1', 59.8, 61.0, 326, 3.89, 3.84, 2.31)
(0.23, 'Good', 'E', 'VS1', 56.9, 65.0, 327, 4.05, 4.07, 2.31)
(0.29, 'Premium', 'I', 'VS2', 62.4, 58.0, 334, 4.2, 4.23, 2.63)
(0.31, 'Good', 'J', 'SI2', 63.3, 58.0, 335, 4.34, 4.35, 2.75)
```

Problem 6: Grouped Means

Out[14]:

	Cut	Count	Mean_Carat	Mean_Price
0	Premium	13791	0.89	4584.26
1	Good	4906	0.85	3928.86
2	Very Good	12082	0.81	3981.76
3	Fair	1610	1.05	4358.76
4	Ideal	21551	0.70	3457.54