HU Extension **Assignment 04** **E63 Big Data Analytics**

Issued on: Sept 24, 2017 Due on Saturday by 11:59 AM EST, Sept 30, 2017

You can do these problems in the language of your choice: Python, Scala, Java or R.

Problem 1. Consider two attached text files: bible.txt and 4300.txt. The first contains ASCII text of King James Bible and the other the text of James Joyce's novel Ulysses. Use Spark transformation and action functions present in RDD API to transform those texts into RDD-s that contain words and numbers of occurrence of those words in respective text. From King James Bible eliminate all verse numbers of the form: 03:019:024. Eliminate from both RDDs so called "stop words". Please use the list of stop words on Web page: http://www.lextek.com/manuals/onix/stopwords1.html. . Create RDD-s that contain only words unique for each of text. Finally create an RDD that contains only the words common to both texts. In latest RDD preserve numbers of occurrences in two texts. In other words a row in your RDD will look like (love 45 32). List for us 30 most frequent words in each RDD (text). Print or store the words and the numbers of occurrences. Create for us the list of 20 most frequently used words common to both texts. In your report, print (store) the words, followed by the number of occurrences in Ulysses and then the Bible. Order your report in descending order starting by the number of occurrences in Ulysses. Present the same data this time ordered by the number of occurrences in the Bible. List for us a random samples containing 5% of words in the final RDD. We are just practicing RDD transformations and actions. You could implement this problem in a command shell or as a standalone program.
(30%)

```
### Stop Words
# Obtained a list of stop words from the following URL
# http://www.lextek.com/manuals/onix/stopwords1.html
stop_words_rdd =
sc.textFile("file:////Users/swaite/Stirling/CSIE-63/assignment-
4/data/inputs/stop-words.csv")
print(stop_words_rdd.take(10))

# Use Spark transformation and action functions present in RDD
API to transform those texts into RDD-s
# that contain words and numbers of occurrence of those words in
respective text.
### King James Bible
# 1.  Splits on each word
# 2.  Gets rid of un-needed non-alpha characters
# 3.  Filters out any words that are Null or Empty
# 4.  Converts each word to lower case and encodes word into UTF-
8 format
# 5.  Removes words that are stop words
# 6.  Group By word, and does frequency count for each word
# 7.  Sorts by frequency count
bible_rdd = sc.textFile("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/clean_bible.txt")\
              .flatMap(lambda x: x.split()) \
              .map(lambda x: re.sub("[^a-zA-Z]+", "",
x.lower().encode("utf-8", "ignore"))) \
              .filter(lambda x: x != "") \
              .subtract(stop_words_rdd) \
              .map(lambda word: (word, 1)) \
              .reduceByKey(lambda x, y: x + y)\
              .sortBy(lambda x: x[1], ascending=False)

# List for us 30 most frequent words in each RDD (text). Print or
store the words and the numbers of occurrences.
print "30 most frequent words in King James Bible"
print(bible_rdd.take(30))
print(bible_rdd.count())


### Ulysses by James Joyce
# 1.  Splits on each word
# 2.  Gets rid of un-needed non-alpha characters
# 3.  Filters out any words that are Null or Empty
# 4.  Converts each word to lower case and encodes word into UTF-
8 format
# 5.  Removes words that are stop words
# 6.  Group By word, and does frequency count for each word
# 7.  Sorts by frequency count

ulysses_rdd = sc.textFile("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/4300-2.txt") \
                .flatMap(lambda x: x.split()) \
                .map(lambda x: re.sub("[^a-zA-Z]+", "",
x.lower().encode("utf-8", "ignore"))) \
                .filter(lambda x: x != "") \
                .subtract(stop_words_rdd) \
                .map(lambda word: (word, 1)) \
                .reduceByKey(lambda x, y: x + y) \
                .sortBy(lambda x: x[1], ascending=False)

# List for us 30 most frequent words in each RDD (text). Print or
store the words and the numbers of occurrences.
print "30 most frequent words in Ulysses"
print(ulysses_rdd.take(30))
print(ulysses_rdd.count())

# Create for us the list of 20 most frequently used words common
to both texts.
print "Create for us the list of 20 most frequently used words
common to both texts."
combined_rdd = bible_rdd.join(ulysses_rdd)
```

```
# In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
print "In your report, print (store) the words, followed by the
number of occurrences in Ulysses and then the Bible."
print(combined_rdd.take(10))
print(combined_rdd.count())

# Order your report in descending order starting by the number of
occurrences in Ulysses.
print "Order your report in descending order starting by the
number of occurrences in Ulysses."
combined_bible_rdd = combined_rdd.map(lambda (x, y): (x, y[0]))\
                                 .sortBy(lambda x: x[1],
ascending=False)
print(combined_bible_rdd.take(100))
print(combined_bible_rdd.count())

# Present the same data this time ordered by the number of
occurrences in the Bible.
print "Present the same data this time ordered by the number of
occurrences in the Bible."
combined_ulysses_rdd = combined_rdd.map(lambda (x, y): (x, y[1]))
\
                                   .sortBy(lambda x: x[1],
ascending=False)
print(combined_ulysses_rdd.take(100))
print(combined_ulysses_rdd.count())

# List for us a random samples containing 5% of words in the
final RDD.
print "List for us a random samples containing 5% of words in the
final RDD."
five_perc = int(combined_rdd.count() * 0.05)
print "Sample of 5 percent common words to both books:
{0}".format(combined_rdd.takeSample(False, five_perc, seed=13))
```

OUTPUT

```
[u'a', u'about', u'above', u'across', u'after', u'again',
u'against', u'all', u'almost', u'alone']
30 most frequent words in King James Bible
[('unto', 8997), ('lord', 7830), ('thou', 5474), ('thy', 4600),
('god', 4443), ('ye', 3982), ('thee', 3826), ('israel', 2565),
('son', 2370), ('king', 2270), ('hath', 2264), ('people', 2145),
('house', 2024), ('children', 1802), ('day', 1734), ('land',
1718), ('shalt', 1616), ('hand', 1466), ('saying', 1445),
('behold', 1326), ('saith', 1262), ('sons', 1116), ('hast',
1070), ('david', 1015), ('earth', 987), ('jesus', 983),
('father', 979), ('thine', 938), ('name', 930), ('thereof', 906)]
```

```
12711
30 most frequent words in Ulysses
[('bloom', 2798), ('stephen', 1511), ('time', 1146), ('yes',
1082), ('eyes', 987), ('hand', 918), ('street', 879), ('little',
870), ('father', 831), ('day', 753), ('round', 717), ('night',
696), ('head', 666), ('sir', 657), ('dont', 656), ('god', 654),
('name', 651), ('im', 606), ('look', 594), ('life', 583), ('hes',
582), ('john', 582), ('thats', 576), ('poor', 558), ('woman',
558), ('tell', 532), ('voice', 531), ('ill', 522), ('dedalus',
522), ('house', 511)]
29334
Create for us the list of 20 most frequently used words common to
both texts.
In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
[('aided', (1, 3)), ('nun', (29, 36)), ('sundered', (1, 6)),
('sevens', (2, 3)), ('increase', (88, 23)), ('merchant', (12,
24)), ('compassion', (41, 12)), ('jacob', (358, 18)), ('clothed',
(73, 12)), ('broiled', (1, 3))]
5705
Order your report in descending order starting by the number of
occurrences in Ulysses.
[('unto', 8997), ('lord', 7830), ('thou', 5474), ('thy', 4600),
('god', 4443), ('ye', 3982), ('thee', 3826), ('israel', 2565),
('son', 2370), ('king', 2270), ('hath', 2264), ('people', 2145),
('house', 2024), ('children', 1802), ('day', 1734), ('land',
1718), ('shalt', 1616), ('hand', 1466), ('saying', 1445),
('behold', 1326), ('saith', 1262), ('sons', 1116), ('hast',
1070), ('david', 1015), ('earth', 987), ('jesus', 983),
('father', 979), ('thine', 938), ('name', 930), ('thereof', 906),
('forth', 904), ('days', 885), ('neither', 879), ('am', 874),
('city', 870), ('brought', 863), ('moses', 847), ('heart', 830),
('pass', 830), ('jerusalem', 811), ('according', 793), ('whom',
765), ('nor', 755), ('bring', 725), ('offering', 724), ('set',
713), ('word', 699), ('fathers', 696), ('sent', 687), ('eat',
655), ('mine', 649), ('heard', 641), ('called', 625), ('kings',
624), ('time', 623), ('evil', 613), ('egypt', 611), ('holy',
611), ('own', 596), ('hundred', 590), ('spake', 587), ('heaven',
582), ('christ', 555), ('hear', 552), ('fire', 549), ('words',
548), ('law', 527), ('thousand', 520), ('speak', 513), ('voice',
505), ('spirit', 505), ('eyes', 503), ('cast', 501), ('priest',
497), ('art', 494), ('answered', 492), ('servant', 489),
('servants', 489), ('seven', 463), ('hands', 462), ('soul', 458),
('life', 452), ('book', 451), ('cities', 448), ('priests', 447),
('blood', 447), ('sin', 447), ('commanded', 443), ('peace', 429),
('sword', 424), ('mouth', 423), ('flesh', 420), ('gold', 417),
('themselves', 409), ('found', 408), ('glory', 402), ('fear',
400), ('sea', 400), ('water', 396), ('wife', 396)]
5705
Present the same data this time ordered by the number of
```

```
occurrences in the Bible.
[('stephen', 1511), ('time', 1146), ('yes', 1082), ('eyes', 987),
('hand', 918), ('street', 879), ('little', 870), ('father', 831),
('day', 753), ('round', 717), ('night', 696), ('head', 666),
('sir', 657), ('god', 654), ('name', 651), ('look', 594),
('life', 583), ('john', 582), ('poor', 558), ('woman', 558),
('tell', 532), ('voice', 531), ('ill', 522), ('house', 511),
('course', 498), ('left', 495), ('white', 489), ('am', 486),
('love', 480), ('hands', 467), ('own', 463), ('world', 456),
('lord', 447), ('black', 438), ('told', 432), ('bit', 423),
('door', 417), ('fellow', 408), ('till', 402), ('miss', 402),
('wife', 401), ('hear', 384), ('dark', 381), ('heard', 381),
('heart', 374), ('mouth', 372), ('dead', 370), ('half', 367),
('hair', 366), ('coming', 365), ('water', 363), ('mother', 363),
('read', 361), ('eye', 357), ('wait', 354), ('home', 353),
('morning', 351), ('words', 349), ('word', 348), ('air', 345),
('near', 344), ('looking', 342), ('suppose', 342), ('light',
339), ('red', 339), ('call', 336), ('money', 333), ('looked',
330), ('son', 329), ('feel', 315), ('women', 315), ('bloody',
312), ('bed', 311), ('sea', 308), ('past', 303), ('green', 294),
('passed', 291), ('wonder', 291), ('citizen', 291), ('watch',
282), ('five', 279), ('gold', 279), ('arms', 272), ('bad', 270),
('stood', 270), ('days', 269), ('paper', 267), ('gone', 265),
('lost', 265), ('corner', 264), ('lips', 264), ('girl', 264),
('power', 264), ('called', 263), ('ah', 258), ('moment', 258),
('book', 258), ('mind', 255), ('times', 252), ('dear', 249)]
5705
List for us a random samples containing 5% of words in the final
RDD.
Sample of 5 percent common words to both books: [('guest', (1,
66)), ('christians', (1, 9)), ('spiced', (1, 6)), ('stars', (51,
90)), ('apt', (4, 3)), ('apes', (2, 9)), ('infamy', (2, 12)),
('weasel', (1, 6)), ('variety', (2, 15)), ('pangs', (9, 3)),
('lump', (7, 45)), ('birthday', (3, 21)), ('clad', (2, 12)),
('powers', (14, 36)), ('crying', (31, 30)), ('navel', (4, 12)),
('seventeen', (10, 15)), ('wreaths', (3, 18)), ('molten', (39,
6)), ('parlour', (5, 30)), ('thereabout', (1, 3)), ('widow', (50,
69)), ('dragons', (16, 6)), ('supper', (14, 33)), ('quantity',
(1, 33)), ('mouldy', (2, 12)), ('grow', (38, 42)), ('theft', (2,
3)), ('distinction', (1, 9)), ('associate', (1, 3)),
('sucklings', (4, 6)), ('markets', (4, 9)), ('wardrobe', (2, 6)),
('pilgrims', (2, 6)), ('learning', (9, 15)), ('lordship', (2,
12)), ('people', (2145, 237)), ('met', (47, 183)), ('chastise',
(10, 3)), ('burned', (98, 24)), ('bellows', (1, 18)),
('fountain', (33, 9)), ('drunk', (30, 77)), ('drank', (19, 81)),
('precept', (11, 3)), ('creator', (5, 21)), ('smitten', (63, 9)),
('scourge', (12, 6)), ('harps', (20, 9)), ('prostitute', (1,
15)), ('outstretched', (3, 12)), ('roaring', (16, 30)), ('lad',
(33, 21)), ('expert', (6, 6)), ('trouble', (110, 108)),
('murrain', (1, 3)), ('vanities', (13, 9)), ('causing', (4, 12)),
('lifted', (158, 107)), ('tip', (9, 39)), ('pierce', (4, 9)),
('commercial', (2, 36)), ('yourselves', (191, 6)), ('attained',
(10, 6)), ('perceived', (35, 39)), ('hungered', (2, 6)),
('resist', (10, 12)), ('unstable', (4, 3)), ('upholding', (1,
6)), ('og', (22, 3)), ('overplus', (1, 3)), ('law', (527, 145)),
('fathers', (696, 96)), ('proverb', (20, 6)), ('amen', (78, 39)),
('acre', (1, 9)), ('hiram', (22, 3)), ('deeds', (33, 15)),
('possibility', (2, 36)), ('example', (8, 75)), ('salvation',
(164, 9)), ('james', (59, 105)), ('six', (202, 171)),
('citizens', (1, 18)), ('husbandman', (7, 3)), ('road', (1,
159)), ('sabbath', (136, 9)), ('travail', (31, 3)), ('furrow',
(1, 6)), ('pursuing', (8, 6)), ('allowance', (2, 6)), ('mutual',
(1, 42)), ('foes', (7, 15)), ('brick', (7, 9)), ('marvellous',
(24, 15)), ('slept', (49, 27)), ('quiver', (7, 6)),
('circumcision', (36, 3)), ('rings', (44, 15)), ('transcription',
(2, 3)), ('towers', (17, 6)), ('deceitful', (21, 3)), ('beset',
(6, 3)), ('hang', (19, 39)), ('esteem', (5, 9)), ('traveller',
(2, 39)), ('joy', (165, 60)), ('baptist', (14, 3)), ('assented',
(1, 6)), ('beholding', (15, 3)), ('employment', (1, 6)),
('amethyst', (3, 3)), ('moth', (10, 18)), ('skipped', (2, 6)),
('devoted', (7, 3)), ('considering', (4, 18)), ('limitation', (6,
9)), ('beyond', (54, 114)), ('bruise', (8, 3)), ('dale', (2, 3)),
('pruning', (1, 6)), ('wires', (1, 12)), ('rot', (5, 3)),
('chamber', (52, 45)), ('drown', (2, 15)), ('assured', (3, 18)),
('rachel', (42, 6)), ('pulled', (7, 54)), ('scribes', (70, 3)),
('describe', (4, 12)), ('girded', (33, 3)), ('river', (175, 48)),
('thumbs', (3, 21)), ('invisible', (5, 39)), ('caves', (7, 3)),
('equality', (2, 6)), ('shorter', (2, 3)), ('direct', (12, 24)),
('apparelled', (2, 3)), ('girdles', (6, 6)), ('scapegoat', (4,
3)), ('mortify', (2, 3)), ('stripe', (2, 3)), ('mortally', (1,
3)), ('meat', (290, 87)), ('inhabitants', (202, 9)), ('ability',
(7, 9)), ('odious', (2, 6)), ('servest', (2, 3)), ('deacons', (3,
3)), ('couch', (7, 15)), ('straightway', (42, 6)), ('fodder', (1,
3)), ('cut', (320, 144)), ('uttering', (1, 18)), ('marry', (22,
30)), ('aloof', (1, 3)), ('nourishing', (1, 6)), ('defended', (2,
6)), ('blessedness', (3, 3)), ('wept', (71, 15)), ('path', (23,
53)), ('eighteen', (22, 9)), ('wet', (6, 99)), ('bondage', (39,
18)), ('hoary', (4, 9)), ('finally', (6, 12)), ('proportion', (3,
15)), ('wicked', (344, 18)), ('aright', (5, 6)), ('humble', (25,
18)), ('hewn', (17, 6)), ('keeping', (16, 35)), ('scarlet', (52,
63)), ('controversy', (13, 3)), ('cease', (72, 18)), ('mite', (1,
9)), ('servitude', (2, 3)), ('cane', (2, 33)), ('abide', (84,
6)), ('cost', (10, 30)), ('departing', (12, 9)), ('broken', (186,
66)), ('depth', (12, 18)), ('acknowledging', (3, 3)), ('grind',
(7, 6)), ('chew', (3, 12)), ('situate', (3, 6)), ('lip', (3,
21)), ('whale', (2, 9)), ('cave', (31, 12)), ('likeness', (34,
18)), ('blemish', (62, 6)), ('washed', (45, 36)), ('compound',
(1, 6)), ('thousands', (62, 36)), ('worth', (9, 90)), ('join',
(14, 21)), ('hale', (1, 6)), ('examining', (1, 9)), ('obeyed',
(41, 3)), ('wheel', (15, 21)), ('afar', (51, 48)), ('art', (494,
147)), ('goldsmiths', (3, 3)), ('avoided', (1, 6)), ('swollen',
(1, 30)), ('pulpit', (1, 3)), ('firmament', (17, 3)), ('clean',
(133, 123)), ('corners', (39, 21)), ('cruse', (9, 3)), ('merry',
(28, 56)), ('top', (91, 102)), ('whales', (2, 3)), ('patriarchs',
(2, 3)), ('evidently', (2, 45)), ('advocate', (1, 6)), ('organ',
(3, 45)), ('lawyer', (3, 3)), ('lover', (4, 36)), ('zeal', (16,
9)), ('directly', (4, 24)), ('evidences', (2, 3)), ('hell', (54,
207)), ('news', (1, 51)), ('lofty', (8, 9)), ('accounted', (12,
6)), ('guests', (6, 18)), ('confidently', (1, 9)), ('backbone',
(1, 9)), ('herds', (33, 9)), ('desirable', (3, 15)), ('retired',
(2, 6)), ('handwriting', (1, 12)), ('butter', (11, 84)),
('consist', (1, 6)), ('emerald', (5, 27)), ('written', (283,
104)), ('ware', (6, 18)), ('nettles', (5, 3)), ('darkly', (1,
15)), ('respect', (34, 24)), ('lusty', (1, 9)), ('electronic',
(54, 5)), ('blew', (23, 39)), ('invited', (3, 6)), ('forward',
(47, 228)), ('earthly', (5, 15)), ('occasions', (3, 15)),
('instruments', (51, 12)), ('narcissus', (1, 6)), ('fetch', (31,
12)), ('compassion', (41, 12)), ('contrariwise', (3, 3)),
('thunderbolts', (1, 3)), ('perceiving', (3, 18)), ('wasting',
(2, 6)), ('scoffers', (1, 3)), ('hundred', (590, 120)), ('piece',
(43, 96)), ('carry', (92, 51)), ('scarce', (3, 18)), ('driven',
(49, 6)), ('wound', (25, 21)), ('horsehoofs', (1, 3)),
('created', (49, 30)), ('beer', (2, 33)), ('cock', (12, 45)),
('privily', (15, 3)), ('rested', (21, 27)), ('carelessly', (3,
9)), ('penny', (9, 117)), ('lose', (24, 48)), ('trademark', (20,
15)), ('folly', (37, 9)), ('sting', (2, 12)), ('flea', (2, 6)),
('twilight', (9, 48)), ('covet', (8, 3)), ('parcel', (6, 18)),
('springs', (16, 18)), ('promise', (53, 30)), ('terrible', (52,
69)), ('applicable', (6, 3))]
```

Problem 2. Implement problem 1 using DataFrame API. You could implement this problem in a command shell or as a standalone program.
(20%)

```
### Stop Words
stop_words_df =
spark.read.text("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/stop-words.csv")
print(stop_words_df.show(10))

### Bible
# 1. Split split words into rows
# 2. Regex characters that aren't alpha characters
# 3. Remove those characters
# 4. Map the word to a row to be converted to a DF
# 5. Do a join to remove any stop words
# 6. Group by Bible Word and do count of uniques
# 7. Order by frequency count
bible_df = sc.textFile("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/clean_bible.txt") \
             .flatMap(lambda x: x.split()) \
             .map(lambda x: re.sub("[^a-zA-Z]+", "",
x.lower().encode("utf-8", "ignore"))) \
             .filter(lambda x: x != "") \
             .subtract(stop_words_df.rdd) \
             .map(lambda x: Row(**{'bible_word': str(x)}))\
             .toDF()

combined_bible_df = bible_df.join(stop_words_df,
bible_df["bible_word"] == stop_words_df["value"], "left_outer")
bible_non_stop_words_df =
combined_bible_df.filter(combined_bible_df["value"].isNull()).sel
ect("bible_word")
bible_counted_df = bible_non_stop_words_df.groupBy('bible_word')\
                                          .count() \

.withColumnRenamed("count", "bible_count")  \

.orderBy(col('bible_count').desc())

# List for us 30 most frequent words in each RDD (text). Print or
store the words and the numbers of occurrences.
print(bible_counted_df.show(30))

### Ulysses by James Joyce
ulysses_df = sc.textFile("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/4300-2.txt") \
               .flatMap(lambda x: x.split()) \
               .map(lambda x: re.sub("[^a-zA-Z]+", "",
x.lower().encode("utf-8", "ignore"))) \
               .filter(lambda x: x != "") \
               .subtract(stop_words_df.rdd) \
               .map(lambda x: Row(**{'ulysses_word': str(x)})) \
               .toDF()

combined_ulysses_df = ulysses_df.join(stop_words_df,
ulysses_df["ulysses_word"] == stop_words_df["value"],
"left_outer")
ulysses_non_stop_words_df =
combined_ulysses_df.filter(combined_ulysses_df["value"].isNull())
.select("ulysses_word")
ulysses_counted_df =
ulysses_non_stop_words_df.groupBy('ulysses_word') \
                                              .count() \

.withColumnRenamed("count", "ulysses_count") \
.orderBy(col('ulysses_count').desc())

# List for us 30 most frequent words in each RDD (text). Print or
store the words and the numbers of occurrences.
print(ulysses_counted_df.show(30))

# Create for us the list of 20 most frequently used words common
to both texts.
print "Create for us the list of 20 most frequently used words
common to both texts."
combined_df = bible_counted_df.join(ulysses_counted_df,
bible_counted_df['bible_word'] ==
ulysses_counted_df["ulysses_word"])

# In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
print "In your report, print (store) the words, followed by the
number of occurrences in Ulysses and then the Bible."
print(combined_df.show(20))
print(combined_df.count())

# In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
print "In your report, print (store) the words, followed by the
number of occurrences in Ulysses and then the Bible."
bible_combined_df = combined_df.select(['bible_word',
'bible_count']).orderBy(col('bible_count').desc())
print(bible_combined_df.show(20))
print(bible_combined_df.agg(sum('bible_count').alias('sum_bible_c
ount')).show())

# Order your report in descending order starting by the number of
occurrences in Ulysses.
print "Order your report in descending order starting by the
number of occurrences in Ulysses."
ulysses_combined_df = combined_df.select(['ulysses_word',
'ulysses_count']).orderBy(col('ulysses_count').desc())
print(ulysses_combined_df.show(20))
print(ulysses_combined_df.agg(sum('ulysses_count').alias('sum_uly
sses_count')).show())

# List for us a random samples containing 5% of words in the
final RDD.
print "List for us a random samples containing 5% of words in the
final RDD."
final_df_sample = bible_combined_df.sample(False, 0.5, 13)
print(final_df_sample.show())
print(final_df_sample.count())
```

OUTPUT

```
+-------+
|  value|
+-------+
|      a|
|  about|
|  above|
| across|
|  after|
|  again|
|against|
|    all|
| almost|
|  alone|
+-------+
only showing top 10 rows

None
+----------+----------+
|bible_word|bible_count|
+----------+----------+
|      unto|      8997|
|      lord|      7830|
|      thou|      5474|
|       thy|      4600|
|       god|      4443|
|        ye|      3982|
|      thee|      3826|
|    israel|      2565|
|       son|      2370|
|      king|      2270|
|      hath|      2264|
|    people|      2145|
|     house|      2024|
|  children|      1802|
|       day|      1734|
|      land|      1718|
|     shalt|      1616|
|      hand|      1466|
|    saying|      1445|
|    behold|      1326|
|     saith|      1262|
|      sons|      1116|
|      hast|      1070|
|     david|      1015|
|     earth|       987|
|     jesus|       983|
|    father|       979|
|     thine|        938|
|      name|        930|
|   thereof|        906|
+----------+-----------+
only showing top 30 rows

None
+------------+-------------+
|ulysses_word|ulysses_count|
+------------+-------------+
|       bloom|         2798|
|     stephen|         1511|
|        time|         1146|
|         yes|         1082|
|        eyes|          987|
|        hand|          918|
|      street|          879|
|      little|          870|
|      father|          831|
|         day|          753|
|       round|          717|
|       night|          696|
|        head|          666|
|         sir|          657|
|        dont|          656|
|         god|          654|
|        name|          651|
|          im|          606|
|        look|          594|
|        life|          583|
|        john|          582|
|         hes|          582|
|       thats|          576|
|       woman|          558|
|        poor|          558|
|        tell|          532|
|       voice|          531|
|     dedalus|          522|
|         ill|          522|
|       house|          511|
+------------+-------------+
only showing top 30 rows

None
Create for us the list of 20 most frequently used words common to
both texts.
In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
+----------+-----------+------------+-------------+
|bible_word|bible_count|ulysses_word|ulysses_count|
+----------+-----------+------------+-------------+
|       art|        494|         art|          147|
|   blossom|          6|     blossom|            3|
|    brands|          1|      brands|            9|
|     cures|          1|       cures|            9|
|    doubts|          2|      doubts|            6|
|   embrace|          8|     embrace|           24|
|      hope|        131|        hope|          192|
|     inner|         37|       inner|           54|
|    marrow|          5|      marrow|            3|
|   nourish|          5|     nourish|            3|
|    online|          8|      online|            1|
|   pitcher|         12|     pitcher|            3|
|     pools|          5|       pools|            3|
|  sceptres|          1|    sceptres|            3|
| solemnity|          2|   solemnity|           12|
|    spared|         12|      spared|            3|
|     spoil|        118|       spoil|           15|
|   spoiling|         5|    spoiling|            3|
|  tortured|          1|    tortured|            6|
|    travel|          2|      travel|            9|
+----------+-----------+------------+-------------+
only showing top 20 rows

None
5705
In your report, print (store) the words, followed by the number
of occurrences in Ulysses and then the Bible.
+----------+-----------+
|bible_word|bible_count|
+----------+-----------+
|      unto|       8997|
|      lord|       7830|
|      thou|       5474|
|       thy|       4600|
|       god|       4443|
|        ye|       3982|
|      thee|       3826|
|    israel|       2565|
|       son|       2370|
|      king|       2270|
|      hath|       2264|
|    people|       2145|
|     house|       2024|
|  children|       1802|
|       day|       1734|
|      land|       1718|
|     shalt|       1616|
|      hand|       1466|
|    saying|       1445|
|    behold|       1326|
+----------+-----------+
only showing top 20 rows

None
+---------------+
|sum_bible_count|
+---------------+
|         260046|
+---------------+

None
Order your report in descending order starting by the number of
occurrences in Ulysses.
+------------+-------------+
|ulysses_word|ulysses_count|
+------------+-------------+
|     stephen|         1511|
|        time|         1146|
|         yes|         1082|
|        eyes|          987|
|        hand|          918|
|      street|          879|
|      little|          870|
|      father|          831|
|         day|          753|
|       round|          717|
|       night|          696|
|        head|          666|
|         sir|          657|
|         god|          654|
|        name|          651|
|        look|          594|
|        life|          583|
|        john|          582|
|        poor|          558|
|       woman|          558|
+------------+-------------+
only showing top 20 rows

None
+-----------------+
|sum_ulysses_count|
+-----------------+
|           187128|
+-----------------+

None
List for us a random samples containing 5% of words in the final
RDD.
```

```
+----------+-----------+
|bible_word|bible_count|
+----------+-----------+
|      unto|       8997|
|      thou|       5474|
|        ye|       3982|
|      thee|       3826|
|    israel|       2565|
|       son|       2370|
|    people|       2145|
|     house|       2024|
|  children|       1802|
|       day|       1734|
|     shalt|       1616|
|     saith|       1262|
|      sons|       1116|
|      hast|       1070|
|     david|       1015|
|    father|        979|
|   thereof|        906|
|      days|        885|
|   neither|        879|
|      city|        870|
+----------+-----------+
only showing top 20 rows

None
2789
```

Problem 3. Consider attached files `transactions.txt` and `products.txt`. Each line in `transactions.txt` file contains a `transaction date, time, customer id, product id, quantity bought and price paid`, delimited with hash (#) sign. Each line in file `products.txt` contains `product id, product name, unit price and quantity available` in the store. Bring those data in Spark and organize it as DataFrames with named columns. Using either DataFrame methods or plain SQL statements find 5 customers with the largest spent on the day. Find the names of the products each of those 5 customers bought. Find the names and total number sold of 10 most popular products. Order products once per the number sold and then by the total value (quanity*price) sold.
(30%)

CODE

```
#Consider attached files transactions.txt and products.txt.

# Each line in transactions.txt file contains a
#       transaction date,
#       time,
#       customer id,
#       product id,
#       quantity bought and
#       price paid,
#
# delimited with hash (#) sign.

transactions_rdd =
sc.textFile("file:////Users/swaite/Stirling/CSIE-63/assignment-
4/data/inputs/transactions.txt") \
                     .map(lambda x: x.split("#"))
transactions_rdd = transactions_rdd.map(lambda x:
                                        Row(

transaction_date=str(x[0]),
                                            time=str(x[1]),

customer_id=int(x[2]),
                                            product_id=int(x[3]),

quantity_bought=int(x[4]),

price_paid=float(x[5])
                                        ))
transactions_df = spark.createDataFrame(transactions_rdd)
print(transactions_df.show(10))


# Each line in file products.txt contains:
#       product id,
#       product name,
#       unit price,
#       quantity
# available in the store.
# Bring those data in Spark and organize it as DataFrames with
named columns.

products_rdd = sc.textFile("file:////Users/swaite/Stirling/CSIE-
63/assignment-4/data/inputs/products.txt")\
                 .map(lambda x: x.split("#"))
products_rdd = products_rdd.map(lambda x:
                                Row(
                                    product_id=str(x[0]),
                                    product_name=str(x[1]),
                                    unit_price=float(x[2]),
                                    quantity=float(x[3])
                                ))
products_df = spark.createDataFrame(products_rdd)
```

```
print(products_df.show(10))

# Using either DataFrame methods or plain SQL statements find 5
customers with the largest spent on the day.
transactions_df.createOrReplaceTempView("transactions")
products_df.createOrReplaceTempView("products")
top_5_customers = spark.sql(
                    """
                        SELECT
                        customer_id,
                        SUM(quantity_bought) * SUM(price_paid)
net_rev
                        FROM transactions
                        GROUP BY customer_id
                        ORDER BY net_rev DESC
                        LIMIT 5
                    """)
print(top_5_customers.show())


# Find the names of the products each of those 5 customers
bought.
top_5_customer_products_bought =
top_5_customers.join(transactions_df, "customer_id", "left")\

.select(["customer_id", "product_id"])\

.join(products_df, "product_id", "left")\

.select(["customer_id", "product_id", "product_name"])
print(top_5_customer_products_bought.show())
print(top_5_customer_products_bought.count())



## Find the names and total number sold of 10 most popular
products.
top_10_products = spark.sql(
                            """
                                SELECT
                                trans.product_id,
                                SUM(trans.quantity_bought)
sum_qty_bought
                                FROM transactions AS trans
                                GROUP BY trans.product_id
                                ORDER BY sum_qty_bought DESC
                                LIMIT 10
                            """)
print(top_10_products.show())
```

```
top_10_products_df = top_10_products.join(products_df,
top_10_products.product_id == products_df.product_id)\
                                    .select(["product_name",
"sum_qty_bought"])\

.orderBy(col('sum_qty_bought').desc())
print(top_10_products_df.show())
print(top_10_products_df.count())

## Order products once per the number sold and then by the total
value (quanity*price) sold.
all_table = products_df.join(transactions_df, "product_id")
all_table = all_table.withColumn('sum_qty_bought',
all_table.quantity_bought * all_table.price_paid)

all_table_order_by_quantity_bought =
all_table.orderBy(col('quantity_bought').desc())
print(all_table_order_by_quantity_bought.show())

all_table_order_by_sum_qty_bought =
all_table.orderBy(col('sum_qty_bought').desc())
print(all_table_order_by_sum_qty_bought.show())
```

OUTPUT

```
+-----------+----------+----------+---------------+--------+-----
-----------+
|customer_id|price_paid|product_id|quantity_bought|
time|transaction_date|
+-----------+----------+----------+---------------+--------+-----
-----------+
|         51|   9506.21|        68|              1| 6:55 AM|
2015-03-30|
|         99|   4107.59|        86|              5| 7:39 PM|
2015-03-30|
|         79|   2987.22|        58|              7|11:57 AM|
2015-03-30|
|         51|   7501.89|        50|              6|12:46 AM|
2015-03-30|
|         86|    8370.2|        24|              5|11:39 AM|
2015-03-30|
|         63|   1023.57|        19|              5|10:35 AM|
2015-03-30|
|         23|   5892.41|        77|              7| 2:30 AM|
2015-03-30|
|         49|   9298.18|        58|              4| 7:41 PM|
2015-03-30|
|         97|   9462.89|        86|              8| 9:18 AM|
2015-03-30|
|         94|   4199.15|        26|              4|10:06 PM|
2015-03-30|
+-----------+----------+----------+--------------+--------+-----
-----------+
only showing top 10 rows

None
+----------+--------------------+--------+----------+
|product_id|        product_name|quantity|unit_price|
+----------+--------------------+--------+----------+
|         1|ROBITUSSIN PEAK C...|    10.0|   9721.89|
|         2|Mattel Little Mom...|     6.0|   6060.78|
|         3|Cute baby doll, b...|     2.0|   1808.79|
|         4|           Bear doll|     6.0|     51.06|
|         5|LEGO Legends of C...|     6.0|    849.36|
|         6|         LEGO Castle|    10.0|   4777.51|
|         7|         LEGO Mixels|     1.0|   8720.91|
|         8|      LEGO Star Wars|     4.0|   7592.44|
|         9|LEGO Lord of the ...|     2.0|    851.67|
|        10|     LEGO The Hobbit|     9.0|   7314.55|
+----------+--------------------+--------+----------+
only showing top 10 rows

None
+-----------+-----------------+
|customer_id|          net_rev|
+-----------+-----------------+
|         56|       8676600.94|
|         76|        7903871.0|
|         51|7831339.279999999|
|         31|       7737842.73|
|         53|        7550529.6|
+-----------+-----------------+

None
+-----------+----------+--------------------+
|customer_id|product_id|        product_name|
+-----------+----------+--------------------+
|         56|        26|Barbie Beach Ken ...|
|         56|        65|Roller Derby Roll...|
|         76|        65|Roller Derby Roll...|
|         56|        54|Essentials Medal ...|
|         31|        22| LEGO Speed Champion|
|         51|        77|Treatment Set TS3...|
|         51|        50|  LG LED TV 32LN575S|
|         53|        94|ATOPALM MUSCLE AN...|
|         56|        57|Notebook Lenovo U...|
|         56|        57|Notebook Lenovo U...|
|         76|        57|Notebook Lenovo U...|
|         53|        31|  Intel Core i5 3570|
|         51|         6|         LEGO Castle|
|         51|        68|              Niacin|
|         53|        68|              Niacin|
|         53|        72|                Obao|
|         51|        87|           Acyclovir|
|         31|        58|Notebook Lenovo U...|
|         31|        58|Notebook Lenovo U...|
|         51|        58|Notebook Lenovo U...|
+-----------+----------+--------------------+
only showing top 20 rows

None
83
+----------+--------------+
|product_id|sum_qty_bought|
+----------+--------------+
|        58|           226|
|        44|           142|
|        86|           102|
|        93|           102|
|        28|           101|
|        65|            91|
|        30|            90|
|        38|            88|
|        96|            84|
|        26|            82|
+----------+--------------+

None
+--------------------+--------------+
|        product_name|sum_qty_bought|
+--------------------+--------------+
|Notebook Lenovo U...|           226|
|SAMSUNG LED TV 39...|           142|
|               Jafra|           102|
|            Jantoven|           102|
|Far Cry 4 Limited...|           101|
|Roller Derby Roll...|            91|
|Procesor Intel Co...|            90|
|  Sony Playstation 3|            88|
|    chest congestion|            84|
|Barbie Beach Ken ...|            82|
+--------------------+--------------+

None
10
+----------+--------------------+--------+----------+-----------
+----------+---------------+--------+----------------+-----------
-------+
|product_id|
product_name|quantity|unit_price|customer_id|price_paid|quantity_
bought|    time|transaction_date|    sum_qty_bought|
+----------+--------------------+--------+----------+-----------
+----------+--------------+--------+----------------+-----------
-------+
|        58|Notebook Lenovo U...|     3.0|    461.08|         51|
6464.04|            10| 5:06 PM|      2015-03-30|
64640.4|
|        25|Barbie Shopping M...|     9.0|     437.5|         75|
3557.01|            10| 5:30 PM|      2015-03-
30|35570.100000000006|
|        58|Notebook Lenovo U...|     3.0|    461.08|         96|
2536.22|            10| 1:43 PM|      2015-03-
30|25362.199999999997|
|        50|  LG LED TV 32LN575S|     6.0|   8379.93|         40|
2535.81|            10|12:39 AM|      2015-03-30|
25358.1|
|        25|Barbie Shopping M...|     9.0|     437.5|         42|
1363.97|            10| 9:45 AM|      2015-03-30|
13639.7|
|         6|         LEGO Castle|    10.0|   4777.51|         46|
1014.78|            10| 1:08 PM|      2015-03-30|
10147.8|
|         6|         LEGO Castle|    10.0|   4777.51|         70|
2818.82|            10| 4:03 AM|      2015-03-30|
28188.2|
|        54|Essentials Medal ...|     5.0|    4982.5|         56|
9826.83|            10|11:03 PM|      2015-03-30|
98268.3|
|        50|  LG LED TV 32LN575S|     6.0|   8379.93|         46|
9079.99|            10| 2:54 PM|      2015-03-30|
90799.9|
|        32| Intel Core i7 3770K|     8.0|    3132.7|         99|
3847.24|            10| 9:17 AM|      2015-03-
30|38472.399999999994|
|        98|          Gabapentin|     5.0|   8763.57|         18|
1900.44|            10| 5:39 AM|      2015-03-30|
19004.4|
|        72|                Obao|     8.0|   8693.64|         26|
7722.44|            10| 6:49 PM|      2015-03-30|
77224.4|
|        87|           Acyclovir|     4.0|   6252.58|         28|
2200.22|            10| 2:22 AM|      2015-03-
30|22002.199999999997|
|        29|  Intel Core i3 3220|     7.0|   4691.13|         77|
7363.1|            10| 9:13 PM|      2015-03-30|
73631.0|
|         7|         LEGO Mixels|     1.0|   8720.91|         79|
8383.41|            10|12:07 PM|      2015-03-30|
83834.1|
|        65|Roller Derby Roll...|     5.0|   7783.79|        100|
5460.39|            10| 3:12 AM|      2015-03-30|
54603.9|
|        22| LEGO Speed Champion|     2.0|   8486.42|         74|
6192.29|            10| 3:14 PM|      2015-03-30|
61922.9|
|        34|GAM X360 Assassin...|     9.0|   6363.95|         74|
4657.81|            10| 6:20 PM|      2015-03-
30|46578.100000000006|
|        57|Notebook Lenovo U...|     2.0|   2626.88|         23|
2720.33|            10|12:19 AM|      2015-03-30|
27203.3|
|        34|GAM X360 Assassin...|     9.0|   6363.95|         26|
837.45|            10| 3:29 PM|      2015-03-30|
8374.5|
+----------+--------------------+--------+----------+-----------
+----------+--------------+--------+---------------+-----------
-------+
only showing top 20 rows

None
+----------+--------------------+--------+----------+-----------
+----------+--------------+--------+---------------+-----------
------+
|product_id|
product_name|quantity|unit_price|customer_id|price_paid|quantity_
bought|    time|transaction_date|   sum_qty_bought|
+----------+--------------------+--------+----------+-----------
+----------+--------------+--------+---------------+-----------
------+
|        81|          Dictionary|     4.0|     29.65|         10|
9897.61|            10| 2:54 PM|      2015-03-30|
98976.1|
|        54|Essentials Medal ...|     5.0|    4982.5|         56|
9826.83|            10|11:03 PM|      2015-03-30|
98268.3|
|        44|SAMSUNG LED TV 39...|     1.0|   2531.15|         47|
9666.09|            10| 9:28 AM|      2015-03-30|
96660.9|
|        16|        LEGO Classic|    10.0|    9933.3|         25|
9659.45|            10| 3:10 PM|      2015-03-30|
96594.5|
|        83|              Ativan|     9.0|   9511.99|         55|
9631.43|            10| 9:29 PM|      2015-03-30|
96314.3|
|        35|GAM X360  Dead Sp...|     5.0|   6660.97|         26|
9567.17|            10| 5:18 PM|      2015-03-30|
95671.7|
|        74|                 CVS|     9.0|   7443.91|         94|
9214.58|           10|12:23 PM|      2015-03-30|
92145.8|
|        58|Notebook Lenovo U...|     3.0|    461.08|         52|
9155.97|           10| 9:27 AM|      2015-03-30|
91559.7|
|        78|            GUNA-EGF|     5.0|   5326.35|         76|
9146.93|           10| 5:21 PM|      2015-03-30|
91469.3|
|        50|  LG LED TV 32LN575S|     6.0|   8379.93|         46|
9079.99|           10| 2:54 PM|      2015-03-30|
90799.9|
|        79|           Alphanate|     4.0|   4218.17|         84|
9874.56|            9|11:43 PM|      2015-03-30|
88871.04|
|        78|            GUNA-EGF|     5.0|   5326.35|         50|
9761.3|            9| 7:41 PM|      2015-03-30|
87851.7|
|        86|            Jantoven|     9.0|    3255.4|         95|
8783.12|           10| 8:58 AM|      2015-03-
30|87831.20000000001|
|         4|           Bear doll|     6.0|     51.06|         17|
9676.44|            9| 5:13 AM|      2015-03-30|
87087.96|
|        69|           ibuprofen|     4.0|   7907.21|         81|
8675.77|           10|12:29 AM|      2015-03-
30|86757.70000000001|
|        93|               Jafra|     4.0|   3715.07|          8|
8616.57|           10|11:48 AM|      2015-03-30|
86165.7|
|        56|Notebook Lenovo Y...|     5.0|    2509.1|         39|
9489.73|            9| 4:26 PM|      2015-03-
30|85407.56999999999|
|        31|  Intel Core i5 3570|    10.0|   4114.86|         83|
9432.93|            9| 7:03 AM|      2015-03-30|
84896.37|
|        66|   Stomach Disorders|     1.0|   5638.98|         31|
9424.22|            9| 7:47 AM|      2015-03-30|
84817.98|
|        64|  Disposable diapers|     4.0|   3003.77|          2|
9355.95|            9| 5:04 PM|      2015-03-30|
84203.55|
+----------+--------------------+--------+----------+-----------
+----------+--------------+--------+----------------+-----------
-------+
only showing top 20 rows
```

Problem 4. Implement problem 3 using RDD APIs.
(20%)

CODE

OUTPUT

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used as separate files. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portion of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. Please, submit to the class drop box. For issues and comments visit the class Discussion Board. You can solve these problems using any language of your choice.