Assignment 4
CSCI E 63 - Big Data Analytics

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.

Answer:

→ Read the contents of the King James Bible and James Joyce's Novel Ulysses after deleting the initial pages that don't belong to the text.

```
>>> bibleLines = sc.textFile("bible.txt")
>>> bibleWordsRDD = bibleLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp).filter(lambda x: ':' not in x)
```

→ 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.

Function to clean up the trailing spaces, punctuations and convert to lower case

```
>>> from string import punctuation
>>> def cleanUp(x):
...     x=x.strip(punctuation).lower().strip()
...     return(x);
...
```

Get the list of words in the document by splitting on spaces, clean up the words,

```
>>> bibleWordCounts = bibleLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp)
```

Assignment 4
CSCI E 63 - Big Data Analytics

Verify the data

```
>>> bibleWordCounts.take(100)
```

[u'book', u'01', u'genesis', u'01:001:001', u'in', u'the', u'beginning', u'god', u'created', u'the', u'heaven', u'and', u'the', u'earth', u'01:001:002', u'and', u'the', u'earth', u'was', u'without', u'form', u'and', u'void', u'and', u'darkness', u'was', u'upon', u'the', u'face', u'of', u'the', u'deep', u'and', u'the', u'spirit', u'of', u'god', u'moved', u'upon', u'the', u'face', u'of', u'the', u'waters', u'01:001:003', u'and', u'god', u'said', u'let', u'there', u'be', u'light', u'and', u'there', u'was', u'light', u'01:001:004', u'and', u'god', u'saw', u'the', u'light', u'that', u'it', u'was', u'good', u'and', u'god', u'divided', u'the', u'light', u'from', u'the', u'darkness', u'01:001:005', u'and', u'god', u'called', u'the', u'light', u'day', u'and', u'the', u'darkness', u'he', u'called', u'night', u'and', u'the', u'evening', u'and', u'the', u'morning', u'were', u'the', u'first', u'day', u'01:001:006', u'and', u'god']

We have a clean set of words exactly as in the text.
Now overwrite this with the words and their corresponding counts.

```
>>> bibleWordCounts = bibleLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp).map(lambda word: (word,1)).reduceByKey(lambda a, b: a + b)
```

Verify the data

```
>>> bibleWordCounts.take(100)
```

[(u'16:010:019', 1), (u'16:010:018', 1), (u'16:010:017', 1), (u'16:010:016', 1), (u'16:010:015', 1), (u'16:010:014', 1), (u'16:010:013', 1), (u'16:010:012', 1), (u'16:010:011', 1), (u'16:010:010', 1), (u'aijalon', 7), (u'hanging', 18), (u'30:009:013', 1), (u'sevens', 2), (u'shammuah', 1), (u'ahiman', 4), (u'49:006:009', 1), (u'49:006:008', 1), (u'49:006:001', 1), (u'49:006:003', 1), (u'49:006:002', 1), (u'49:006:005', 1), (u'49:006:004', 1), (u'49:006:007', 1), (u'49:006:006', 1), (u'04:004:024', 1), (u'04:004:025', 1), (u'04:004:026', 1), (u'04:004:027', 1), (u'04:004:020', 1), (u'04:004:021', 1), (u'04:004:022', 1), (u'04:004:023', 1), (u'26:009:007', 1), (u'26:009:006', 1), (u'26:009:005', 1), (u'26:009:004', 1), (u'04:004:028', 1), (u'04:004:029', 1), (u'26:009:001', 1), (u'bringing', 24), (u'four', 328), (u'09:017:026', 1), (u'broiled', 1), (u'woods', 1), (u'09:017:022', 1), (u'12:013:024', 1), (u'complainers', 1), (u'43:018:019', 1), (u'43:018:018', 1), (u'43:018:015', 1), (u'43:018:014', 1), (u'43:018:017', 1), (u'scraped', 2), (u'43:018:011', 1), (u'43:018:010', 1), (u'43:018:013', 1), (u'43:018:012', 1), (u'errors', 5), (u'19:080:008', 1), (u'19:080:009', 1), (u'19:080:001', 1), (u'19:080:002', 1), (u'19:080:003', 1), (u'19:080:004', 1), (u'19:080:005', 1), (u'19:080:006', 1), (u'shocks', 1), (u'crouch', 1), (u'19:147:012', 1), (u'26:030:003', 1), (u'26:030:002', 1), (u'26:030:001', 1), (u'kids', 8), (u'23:011:009', 1), (u'23:011:008', 1), (u'26:030:005', 1), (u'26:030:004', 1), (u'23:011:005', 1), (u'mahalah', 1), (u'23:011:007', 1), (u'climbed', 2), (u'23:011:001', 1), (u'33:005:010', 1), (u'23:011:003', 1), (u'controversy', 13), (u'33:005:011', 1), (u'golden', 66), (u'33:005:014', 1), (u'33:005:015', 1), (u'lengthen', 2), (u'09:025:008', 1), (u'09:025:009', 1), (u'satest', 2), (u'forborn', 1), (u'09:025:001', 1), (u'09:025:002', 1), (u'09:025:003', 1), (u'09:025:004', 1)]

Assignment 4
CSCI E 63 - Big Data Analytics

Repeat the same steps for the Ulysses text

```
>>> ulyssesLines = sc.textFile("4300-2.txt")
>>> ulyssesWordsRDD = ulyssesLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp).filter(lambda x: ':' not in x)
>>> ulyssesWordCounts = ulyssesLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp)
>>> ulyssesWordCounts.take(100)
[u'ulysses', u'by', u'james', u'joyce', u'', u'', u'stately', u'plump', u'buck', u'mulligan', u'came', u'from', u'the', u'stairhead', u'bearing', u'bowl', u'of', u'lather', u'on', u'which', u'mirror', u'and', u'razor', u'lay', u'crossed', u'yellow', u'dressinggown', u'ungirdled', u'was', u'sustained', u'gently', u'behind', u'him', u'on', u'the', u'mild', u'morning', u'air', u'he', u'held', u'the', u'bowl', u'aloft', u'and', u'intoned', u'introibo', u'ad', u'altare', u'dei', u'halted', u'he', u'peered', u'down', u'the', u'dark', u'winding', u'stairs', u'and', u'called', u'out', u'coarsely', u'come', u'up', u'kinch', u'come', u'up', u'you', u'fearful', u'jesuit', u'solemnly', u'he', u'came', u'forward', u'and', u'mounted', u'the', u'round', u'gunrest', u'he', u'faced', u'about', u'and', u'blessed', u'gravely', u'thrice', u'the', u'tower', u'the', u'surrounding', u'land', u'and', u'the', u'awaking', u'mountains', u'then', u'catching', u'sight', u'of', u'stephen', u'dedalus']

>>> ulyssesWordCounts = ulyssesLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp).map(lambda word: (word,1)).reduceByKey(lambda a, b: a + b)

>>> ulyssesWordCounts.take(100)
[(u'fawn', 9), (u'highspliced', 3), (u'piffpaff', 3), (u'askew', 12), (u'woods', 15), (u'clotted', 12), (u'phenomenologist', 3), (u'hanging', 86), (u'noctambules', 3), (u'comically', 3), (u'houyhnhnm', 3), (u'sevens', 3), (u'canes', 3), (u'sprague', 3), (u'scutter', 3), (u'originality', 6), (u'alphabetic', 3), (u'stipulate', 3), (u'pigment', 3), (u'fullblooded', 3), (u'bringing', 51), (u'four', 282), (u'liaisons', 3), (u'wooden', 18), (u'wednesday', 3), (u'virtuosos', 3), (u'broiled', 3), (u'agnathia', 3), (u'bullswords', 3), (u'sooty', 3), (u'jamjam', 3), (u'insular', 3), (u'splendiferous', 3), (u'ffoo', 3), (u'sooth', 6), (u'gorman', 6), (u'sustaining', 6), (u'consenting', 3), (u'279', 3), (u'scraped', 21), (u'errors', 20), (u'semicircular', 3), (u'cooking', 9), (u'slabbed', 3), (u'designing', 3), (u'pawed', 12), (u'shocks', 6), (u"hengler's", 9), (u'sexus', 3), (u'perfunctorily', 3), (u'china', 24), (u'affiliated', 3), (u'chink', 6), (u'doldrums', 3), (u'kids', 15), (u'gruntlings', 3), (u'climbed', 15), (u'horseshow', 3), (u'natures', 6), (u'tootling', 3), (u'golden', 63), (u'projection', 3), (u'stern', 13), (u'blumenbach', 3), (u"lapidary's", 3), (u'catchy', 3), (u'insecurity', 3), (u'cannibal', 3), (u'pettiwidth', 3), (u'hazeleyes', 3), (u'music', 228), (u'therefore', 30), (u'finucane', 3), (u'nighthag', 3), (u'heatless', 3), (u'selfabuse', 3), (u'primeval', 6), (u'schoolboys', 24), (u'circumstances', 18), (u'morally', 3), (u'locked', 36), (u'talboys', 21), (u'nailfile', 3), (u'ungyved', 3), (u'locker', 15), (u'darkmans', 3), (u'locket', 6), (u"l'isle", 3), (u'wand', 15), (u'wane', 3), (u'unjust', 3), (u'raimeis', 3), (u'want', 457), (u'absolute', 15), (u'glasseyes', 3), (u'spellingbee', 3), (u'impotable', 3), (u'travel', 9), (u'copious', 3)]
```

Assignment 4
CSCI E 63 - Big Data Analytics

→ From King James Bible eliminate all verse numbers of the form: 03:019:024.

```
>>> bibleWordCounts = bibleLines.flatMap(lambda line: line.split(" ")).filter(lambda x: len(x) > 1).map(cleanUp).map(lambda word: (word,1)).reduceByKey(lambda a, b: a + b).filter(lambda x: ':' not in x[0])
```

Verify that the RDD now looks like the one before but without the verses

```
>>> bibleWordCounts.take(100)
```

[(u'aijalon', 7), (u'hanging', 18), (u'sevens', 2), (u'shammuah', 1), (u'ahiman', 4), (u'bringing', 24), (u'four', 328), (u'broiled', 1), (u'woods', 1), (u'complainers', 1), (u'scraped', 2), (u'errors', 5), (u'shocks', 1), (u'crouch', 1), (u'kids', 8), (u'mahalah', 1), (u'climbed', 2), (u'controversy', 13), (u'golden', 66), (u'lengthen', 2), (u'satest', 2), (u'forborn', 1), (u'therefore', 1237), (u'fatling', 1), (u'rescueth', 1), (u'locked', 2), (u'melchishua', 2), (u'gershom', 14), (u'gershon', 18), (u'ziphion', 1), (u'unjust', 17), (u'want', 31), (u'slothful', 15), (u'travel', 2), (u'abihu', 12), (u'acknowledgeth', 1), (u'jeribai', 1), (u'wrong', 26), (u'twined', 21), (u'rewarder', 1), (u'rewarded', 14), (u'mahazioth', 2), (u'fir', 21), (u'wickedly', 23), (u'fit', 9), (u'fix', 1), (u'uriah', 27), (u'jeaterai', 1), (u'fig', 41), (u'rezeph', 2), (u'raiseth', 8), (u'graving', 3), (u'bethanoth', 1), (u'sixteen', 23), (u'troublous', 1), (u'arrow', 16), (u'burial', 6), (u'ha', 2), (u'encourage', 4), (u'estimate', 2), (u'purtenance', 1), (u'clamorous', 1), (u'service', 133), (u'reuben', 74), (u'master', 157), (u'genesis', 2), (u'rewards', 5), (u'nephtoah', 2), (u'cisterns', 2), (u'jimnites', 1), (u'asaph', 33), (u'idle', 11), (u'feeling', 2), (u'valiant', 32), (u'goodness', 51), (u'plenteousness', 2), (u'remmon', 1), (u'affairs', 8), (u'wholesome', 2), (u'zelzah', 1), (u'appertain', 2), (u'committing', 2), (u'diminishing', 1), (u'joshaviah', 1), (u'mouth', 423), (u'reverence', 13), (u'expound', 1), (u'singer', 2), (u'singed', 1), (u'ahlab', 1), (u'cyrene', 4), (u'magpiash', 1), (u'purged', 14), (u'ahlai', 2), (u'saying', 1445), (u'teresh', 2), (u'tempter', 2), (u'tempted', 25), (u'apace', 3)]

→ 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 a text file with the words from the web page(StopWords.txt)

```
-rw-r--r--. 1 kbhandarkar kbhandarkar    2914 Sep 28 23:34 StopWords.txt
```

Read it into a RDD

```
>>> stopWordsRDD = sc.textFile('StopWords.txt')
```

Remove the words in stopWordRDD from bibleWordCounts

```
>>> stoppedBibleWordCounts = bibleWordCounts.subtract(stopWordsRDD)
>>> stoppedUlyssesWordCounts = ulyssesWordCounts.subtract(stopWordsRDD)
```

→ Create RDD-s that contain only words unique for each of text.

We have our two pair transformed RDDs as bibleWordCounts and ulyssesWordCounts.

Assignment 4
CSCI E 63 - Big Data Analytics

```
>>> bibleWordCounts.take(30)
[(u'aijalon', 7), (u'hanging', 18), (u'sevens', 2), (u'shammuah', 1), (u'ahiman', 4), (u'bringing', 24), (u'four', 328), (u'broiled', 1), (u'woods', 1), (u'complainers', 1), (u'scraped', 2), (u'errors', 5), (u'shocks', 1), (u'crouch', 1), (u'kids', 8), (u'mahalah', 1), (u'climbed', 2), (u'controversy', 13), (u'golden', 66), (u'lengthen', 2), (u'satest', 2), (u'forborn', 1), (u'therefore', 1237), (u'fatling', 1), (u'rescueth', 1), (u'locked', 2), (u'melchishua', 2), (u'gershom', 14), (u'gershon', 18)]

>>> ulyssesWordCounts.take(30)
[(u'fawn', 9), (u'highspliced', 3), (u'piffpaff', 3), (u'askew', 12), (u'woods', 15), (u'clotted', 12), (u'phenomenologist', 3), (u'hanging', 86), (u'noctambules', 3), (u'comically', 3), (u'houyhnhnm', 3), (u'sevens', 3), (u'canes', 3), (u'sprague', 3), (u'scutter', 3), (u'originality', 6), (u'alphabetic', 3), (u'stipulate', 3), (u'pigment', 3), (u'fullblooded', 3), (u'bringing', 51), (u'four', 282), (u'liaisons', 3), (u'wooden', 18), (u'wednesday', 3), (u'virtuosos', 3), (u'broiled', 3), (u'agnathia', 3), (u'bullswords', 3)]
```

Create a RDD with keys common to both the RDDs

```
>>> common = bibleWordCounts.join(ulyssesWordCounts)
>>> common.take(30)
[(u'hanging', (18, 86)), (u'sevens', (2, 3)), (u'bringing', (24, 51)), (u'four', (328, 282)), (u'broiled', (1, 3)), (u'woods', (1, 15)), (u'consenting', (2, 3)), (u'scraped', (2, 21)), (u'errors', (5, 20)), (u'shocks', (1, 6)), (u'kids', (8, 15)), (u'climbed', (2, 15)), (u'controversy', (13, 3)), (u'golden', (66, 63)), (u'therefore', (1237, 30)), (u'harmless', (4, 8)), (u'locked', (2, 36)), (u'unjust', (17, 3)), (u'want', (31, 457)), (u'travel', (2, 9)), (u'twined', (21, 9)), (u'rewarded', (14, 6)), (u'fir', (21, 6)), (u'fit', (9, 54)), (u'sixteen', (23, 36)), (u'burial', (6, 21)), (u'encourage', (4, 10)), (u'estimate', (2, 5)), (u'stripe', (2, 3))]
```

Remove the keys in this RDD from the keys in the two RDDs to leave keys distinct to each RDD

```
>>> uniqueBibleWords = bibleWordCounts.subtractByKey(common)
>>> uniqueBibleWords.take(30)
[(u'giddel', 4), (u'writings', 1), (u'mozah', 1), (u'aijalon', 7), (u'railing', 4), (u'deserveth', 1), (u'cyprus', 8), (u'shammuah', 1), (u'taanathshiloh', 1), (u'meadows', 1), (u'forfeited', 1), (u'treasuries', 10), (u'slothful', 15), (u'hough', 1), (u'rashly', 1), (u'intercessor', 1), (u'scoffers', 1), (u'eshek', 1), (u'us-ascii', 1), (u'chesed', 1), (u'bethanoth', 1), (u'elihu', 11), (u'lionlike', 2), (u'ahinoam', 7), (u'inheritances', 1), (u'hanan', 12), (u'lovest', 12), (u'ligure', 2), (u'complainers', 1), (u'suppliants', 1)]

>>> uniqueUlyssesWords = ulyssesWordCounts.subtractByKey(common)
>>> uniqueUlyssesWords.take(30)
```

Karan A. Bhandarkar

[(u'fawn', 9), (u'highspliced', 3), (u'piffpaff', 3), (u'askew', 12), (u'clotted', 12), (u'phenomenologist', 3), (u'noctambules', 3), (u'comically', 3), (u'houyhnhnm', 3), (u'canes', 3), (u'sprague', 3), (u'scutter', 3), (u'originality', 6), (u'alphabetic', 3), (u'stipulate', 3), (u'pigment', 3), (u'fullblooded', 3), (u'liaisons', 3), (u'wooden', 18), (u'wednesday', 3), (u'virtuosos', 3), (u'agnathia', 3), (u'bullswords', 3), (u'sooty', 3), (u'jamjam', 3), (u'insular', 3), (u'splendiferous', 3), (u'ffoo', 3), (u'sooth', 6), (u'gorman', 6)]

→ List for us 30 most frequent words in each RDD

```
>>> top30BibleWords = uniqueBibleWords.takeOrdered(30, key = lambda x: -x[1])
>>> print top30BibleWords
```
[(u'judah', 812), (u'brethren', 563), (u'iniquity', 278), (u'offerings', 265), (u'destroy', 263), (u'philistines', 254), (u'inheritance', 239), (u'righteous', 238), (u'joshua', 218), (u'cubits', 213), (u'slew', 196), (u'rejoice', 194), (u'commandment', 177), (u'moab', 168), (u'hearken', 153), (u'desolate', 148), (u'manasseh', 143), (u'joab', 137), (u'sanctuary', 137), (u'goeth', 135), (u"lord's", 134), (u'statutes', 132), (u'hezekiah', 128), (u'maketh', 126), (u'smite', 125), (u'samaria', 124), (u'captains', 119), (u'assyria', 118), (u'jonathan', 118), (u'mayest', 114)]

```
>>> top30UlyssesWords = uniqueUlyssesWords.takeOrdered(30, key = lambda x: -x[1])
>>> print top30UlyssesWords
```
[(u'bloom', 2798), (u'mr', 2154), (u'says', 1419), (u'o', 714), (u'mrs', 609), (u"don't", 559), (u'j', 549), (u'dedalus', 522), (u'hat', 504), (u"it's", 468), (u"that's", 468), (u"i'm", 456), (u'mulligan', 450), (u"he's", 423), (u'joe', 411), (u'big', 366), (u'towards', 360), (u'irish', 351), (u'dublin', 348), (u"i'll", 348), (u'buck', 339), (u'martin', 318), (u'zoe', 309), (u'lenehan', 306), (u'gentleman', 263), (u"there's", 255), (u'ireland', 245), (u'henry', 237), (u'nice', 234), (u"didn't", 231)]

→ Create for us the list of 20 most frequently used words common to both texts.

```
>>> summedCommon = common.map(lambda(x,(a,b)):(x,a+b))
>>> summedCommon.take(20)
```
[(u'hanging', 104), (u'sevens', 5), (u'bringing', 75), (u'four', 610), (u'broiled', 4), (u'woods', 16), (u'consenting', 5), (u'scraped', 23), (u'errors', 25), (u'shocks', 7), (u'kids', 23), (u'climbed', 17), (u'controversy', 16), (u'golden', 129), (u'therefore', 1267), (u'harmless', 12), (u'locked', 38), (u'unjust', 20), (u'want', 488), (u'travel', 11)]

→ In your report, print (store) the words, followed by the number of occurrences in Ulysses and then the Bible.

```
>>> commonub = common.map(lambda(x,(b,u)):(x,u,b))
>>> commonub.take(3)
```

Assignment 4
CSCI E 63 - Big Data Analytics

[(u'hanging', 86, 18), (u'sevens', 3, 2), (u'bringing', 51, 24), (u'four', 282, 328), (u'broiled', 3, 1), (u'woods', 15, 1), (u'consenting', 3, 2), (u'scraped', 21, 2), (u'errors', 20, 5), (u'shocks', 6, 1), (u'kids', 15, 8), (u'climbed', 15, 2), (u'controversy', 3, 13), (u'golden', 63, 66), (u'therefore', 30, 1237), (u'harmless', 8, 4), (u'locked', 36, 2), (u'unjust', 3, 17), (u'want', 457, 31), (u'travel', 9, 2), (u'twined', 9, 21), (u'rewarded', 6, 14), (u'fir', 6, 21), (u'fit', 54, 9), (u'sixteen', 36, 23), (u'burial', 21, 6), (u'encourage', 10, 4), (u'estimate', 5, 2), (u'stripe', 3, 2), (u'lightened', 3, 5)]Validated in the document that 'sevens' is present 3 times in Ulysses and 2 times in the Bible.

→ Order your report in descending order starting by the number of occurrences in Ulysses.

```
>>> commonDescUlysses = commonub.sortBy(lambda (a,b,c): b, ascending = False)
>>> commonDescUlysses.take(10)
```
[(u'the', 44954, 64109), (u'of', 24516, 34743), (u'and', 21714, 51766), (u'to', 15023, 13642), (u'in', 14806, 12726), (u'he', 12080, 10422), (u'his', 9983, 8473), (u'that', 7835, 12928), (u'with', 7554, 6061), (u'it', 7110, 6144)]

→Present the same data this time ordered by the number of occurrences in the Bible.

```
>>> commonDescBible = commonub.sortBy(lambda (a,b,c): c, ascending = False)
>>> commonDescBible.take(10)
```
[(u'the', 44954, 64109), (u'and', 21714, 51766), (u'of', 24516, 34743), (u'to', 15023, 13642), (u'that', 7835, 12928), (u'in', 14806, 12726), (u'he', 12080, 10422), (u'shall', 198, 9840), (u'for', 5845, 8997), (u'unto', 15, 8997)]

→ List for us a random samples containing 5% of words in the final RDD.

Since it is not specified, I have assumed that the RDD with the words common to Ulysses and Bible is the 'final' RDD.

```
>>> common.count()
6098
>>> sampleCommon = common.sample(False,0.05)
>>> sampleCommon.count()
303
```
a.sample(false, 0.1) internally uses Bernaoulli sampling so doesn't return the same on every call. If we want an exact sample size i.e. exactly 5% of words

```
>>> sampleCommon = common.takeSample(False,305)
```
The 305 is 5% of the total count. This returns an exact count but in the form of a list.

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

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

Answer:

→ Read the contents of the King James Bible and James Joyce's Novel Ulysses after deleting the initial pages that don't belong to the text.

Define a function to clean up the text

```
>>> from pyspark.sql.functions import regexp_replace, trim, col, lower
>>> def removePunctuation(column):
...     return trim(lower(regexp_replace(column, '[^\sa-zA-Z0-9]', ''))).alias('sentence')
...
```

Read the Ulysses text and clean up the text

```
>>> ulyssesDF = sqlContext.read.text("4300-2.txt").select(removePunctuation(col('value')))
>>> ulyssesDF.show(15)
```

```
+--------------------+
|            sentence|
+--------------------+
|ulysses by james ...|
|                    |
|                    |
|                    |
|                    |
|                    |
|                   i|
|                    |
|                    |
|                    |
|                    |
|                    |
|                    |
|                    |
|stately plump buc...|
|                    |
|lather on which a...|
|                    |
|ungirdled was sus...|
|                    |
+--------------------+
only showing top 20 rows
```

Split the text into a DF of words

```
>>> from pyspark.sql.functions import split, explode
>>> ulyssesWordsDF = (ulyssesDF.select(explode(split(ulyssesDF.sentence, '
')).alias('word')).where(col('word') != ''))
>>> ulyssesWordsDF.show()
```

Assignment 4
CSCI E 63 - Big Data Analytics

```
+---------+
|     word|
+---------+
|  ulysses|
|       by|
|    james|
|    joyce|
|        i|
|  stately|
|    plump|
|     buck|
| mulligan|
|     came|
|     from|
|      the|
|stairhead|
|  bearing|
|        a|
|     bowl|
|       of|
|   lather|
|       on|
|    which|
+---------+
only showing top 20 rows
```

Do the same for the bible text

Read the Ulysses text and clean up the text

```
>>> bibleDF = sqlContext.read.text("bible.txt").select(removePunctuation(col('value')))
>>> bibleDF.show(15)
```

```
+--------------------+
|            sentence|
+--------------------+
|book 01        ge...|
|                    |
|                    |
|                    |
|01001001 in the b...|
|                    |
|                    |
|                    |
|01001002 and the ...|
|                    |
|upon the face of ...|
|                    |
|the face of the w...|
|                    |
|                    |
+--------------------+
only showing top 15 rows
```

Split the text into a DF of words

```
>>> bibleWordsDF = (bibleDF.select(explode(split(bibleDF.sentence, '
')).alias('word')).where(col('word') != ''))
>>> bibleWordsDF.show(15)
```

Assignment 4
CSCI E 63 - Big Data Analytics

```
+---------+
|     word|
+---------+
|     book|
|       01|
|  genesis|
| 01001001|
|       in|
|      the|
|beginning|
|      god|
|  created|
|      the|
|   heaven|
|      and|
|      the|
|    earth|
| 01001002|
+---------+
only showing top 15 rows
```

→ Count the words

Define a function to get the word count from a word list

```
>>> def wordCount(wordListDF):
...     return wordListDF.groupBy('word').count()
...
```

Create a dataframe with word and its count in descending order

```
>>> from pyspark.sql.functions import desc
>>> ulyssesWordsAndCountsDF = wordCount(ulyssesWordsDF).orderBy("count", ascending=False)
>>> ulyssesWordsAndCountsDF.show()
```

```
+----+-----+
|word|count|
+----+-----+
| the|44954|
|  of|24516|
| and|21714|
|   a|19539|
|  to|15023|
|  in|14806|
|  he|12080|
| his| 9983|
|   i| 8093|
|that| 7835|
|with| 7554|
|  it| 7110|
| was| 6397|
|  on| 6349|
| for| 5845|
| you| 5782|
| her| 5353|
| him| 4570|
|  is| 4346|
| all| 3978|
+----+-----+
only showing top 20 rows
```

Assignment 4
CSCI E 63 - Big Data Analytics

```
>>> bibleWordsAndCountsDF = wordCount(bibleWordsDF).orderBy("count", ascending=False)
>>> bibleWordsAndCountsDF.show()
```

```
+-----+-----+
| word|count|
+-----+-----+
|  the|64109|
|  and|51766|
|   of|34743|
|   to|13642|
| that|12928|
|   in|12726|
|   he|10422|
|shall| 9840|
|  for| 8997|
| unto| 8997|
|    i| 8854|
|  his| 8473|
|    a| 8234|
| lord| 7830|
| they| 7379|
|   be| 7032|
|   is| 7015|
|  him| 6659|
|  not| 6617|
| them| 6430|
+-----+-----+
only showing top 20 rows
```

→ Eliminate all verse numbers from the bible

Filter out the words which represent the verse. Verses all have a 0 in them.

```
>>> noVerseBibleDF = bibleWordsAndCountsDF.where(~col('word').like("%0%"))
```

→ Eliminate the stop-words

Create a DF of Stop Words

```
>>> stopWordsDF = spark.read.text("StopWords.txt")
>>> stopWordsDF.show(10)
```

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

Assignment 4
CSCI E 63 - Big Data Analytics

Create a temporary view of this DF to query against and exclude
>>> stopWordsDF.createOrReplaceTempView('stopWordsVW')
>>> noStopAndVerseBibleDF = noVerseBibleDF.where('word NOT IN (SELECT value FROM stopWordsVW)')
>>> noStopAndVerseBibleDF.show()

>>> noStopUlyssesDF = ulyssesWordsAndCountsDF.where('word NOT IN (SELECT value FROM stopWordsVW)')
>>> noStopUlyssesDF.show()

```
+--------+-----+
|    word|count|
+--------+-----+
|    unto| 8997|
|    lord| 7830|
|    thou| 5474|
|     thy| 4600|
|     god| 4443|
|      ye| 3982|
|    thee| 3826|
|  israel| 2565|
|     son| 2370|
|    hath| 2264|
|    king| 2263|
|  people| 2142|
|   house| 2024|
|children| 1802|
|     day| 1734|
|    land| 1718|
|   shalt| 1616|
|    hand| 1466|
|  saying| 1445|
|  behold| 1326|
+--------+-----+
only showing top 20 rows
```

```
+-------+-----+
|   word|count|
+-------+-----+
|  bloom| 2798|
|stephen| 1511|
|   time| 1141|
|    yes| 1081|
|   eyes|  987|
|   hand|  918|
| street|  879|
| little|  870|
| father|  831|
|    day|  751|
|  round|  717|
|  night|  696|
|   head|  666|
|    sir|  657|
|    god|  654|
|   dont|  652|
|   name|  651|
|     im|  606|
|   look|  594|
|   life|  583|
+-------+-----+
only showing top 20 rows
```

You can see, as compared to BibleWordsAndCountsDF, stop words such as 'the, and, of, to, that, in' have been eliminated.

→ Create Dataframes that contain only words unique for each text

Create a view of the two data frames to query against
>>> noStopAndVerseBibleDF.createOrReplaceTempView('noStopAndVerseBibleVW')
>>> noStopUlyssesDF.createOrReplaceTempView('noStopUlyssesVW')
Write a query to exclude words in one view from the other DF and vice versa
>>> bibleUniqueWordsDF = noStopAndVerseBibleDF.where('word NOT IN (SELECT word FROM noStopUlyssesVW)')
>>> bibleUniqueWordsDF.show()
>>> ulyssesUniqueWordsDF = noStopUlyssesDF.where('word NOT IN (SELECT word FROM noStopAndVerseBibleVW)')
>>> bibleUniqueWordsDF.show()

Assignment 4
CSCI E 63 - Big Data Analytics

```
+-----------+-----+
|       word|count|
+-----------+-----+
|      judah|  812|
|   brethren|  563|
|   iniquity|  278|
|  offerings|  265|
|    destroy|  263|
|philistines|  254|
|inheritance|  239|
|  righteous|  238|
|     joshua|  218|
|     cubits|  213|
|       slew|  196|
|    rejoice|  194|
|commandment|  177|
|        moab|  168|
|    hearken|  153|
|   desolate|  148|
|   manasseh|  143|
|       joab|  137|
|  sanctuary|  137|
|      goeth|  135|
+-----------+-----+
only showing top 20 rows
```

```
+---------+-----+
|     word|count|
+---------+-----+
|    bloom| 2798|
|     dont|  652|
|       im|  606|
|      hes|  582|
|    thats|  576|
|  dedalus|  522|
|      hat|  504|
| mulligan|  450|
|      joe|  411|
|  towards|  360|
|    irish|  351|
|   dublin|  348|
|    didnt|  342|
|     buck|  339|
|   theres|  333|
|   martin|  318|
|      zoe|  309|
|  lenehan|  306|
|  wouldnt|  294|
|gentleman|  263|
+---------+-----+
only showing top 20 rows
```

→ Create a Dataframe that only contains words common to both texts. Change the alias of the column count for clarity.

>>> commonWordsDF = noStopUlyssesDF.join(noStopAndVerseBibleDF, noStopUlyssesDF['word'] == noStopAndVerseBibleDF['word']).select(noStopAndVerseBibleDF['word'],noStopUlyssesDF['count'].alias('uCount'),noStopAndVerseBibleDF['count'].alias('bCount'))
>>> commonWordsDF.show()

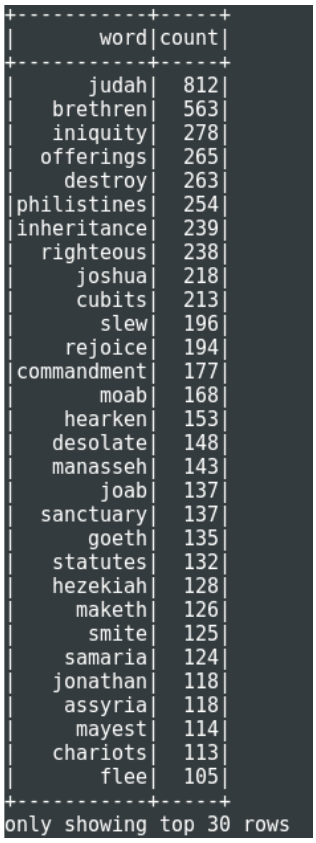
```
+-------+------+------+
|   word|uCount|bCount|
+-------+------+------+
|stephen|  1511|     7|
|   time|  1141|   623|
|    yes|  1081|     4|
|   eyes|   987|   503|
|   hand|   918|  1466|
| street|   879|    36|
| little|   870|   242|
| father|   831|   979|
|    day|   751|  1734|
|  round|   717|   320|
|  night|   696|   307|
|   head|   666|   364|
|    sir|   657|    12|
|    god|   654|  4443|
|   name|   651|   929|
|   look|   594|   155|
|   life|   583|   451|
|   john|   582|   139|
|  woman|   558|   357|
|   poor|   558|   205|
+-------+------+------+
only showing top 20 rows
```

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

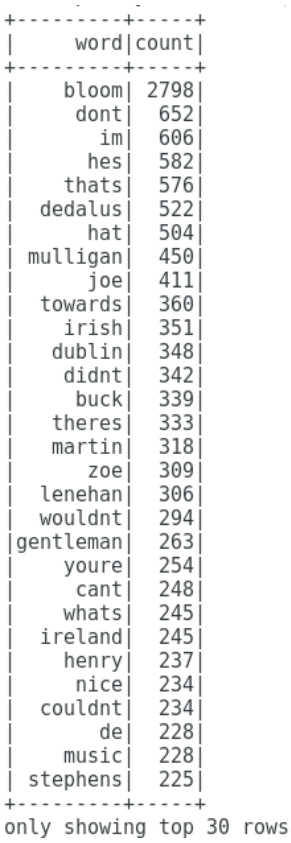
→ List 30 most frequent words in each dataframe

>>> top30BibleDF = bibleUniqueWordsDF.sort("count", ascending = False)
>>> top30UlyssesDF = ulyssesUniqueWordsDF.sort("count", ascending = False)



word	count
judah	812
brethren	563
iniquity	278
offerings	265
destroy	263
philistines	254
inheritance	239
righteous	238
joshua	218
cubits	213
slew	196
rejoice	194
commandment	177
moab	168
hearken	153
desolate	148
manasseh	143
joab	137
sanctuary	137
goeth	135
statutes	132
hezekiah	128
maketh	126
smite	125
samaria	124
jonathan	118
assyria	118
mayest	114
chariots	113
flee	105

only showing top 30 rows



word	count
bloom	2798
dont	652
im	606
hes	582
thats	576
dedalus	522
hat	504
mulligan	450
joe	411
towards	360
irish	351
dublin	348
didnt	342
buck	339
theres	333
martin	318
zoe	309
lenehan	306
wouldnt	294
gentleman	263
youre	254
cant	248
whats	245
ireland	245
henry	237
nice	234
couldnt	234
de	228
music	228
stephens	225

only showing top 30 rows

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

→ Create a list of 20 most frequently used words common to both the texts. Print (store) the words, followed by the number of occurrences in Ulysses and then the Bible.

>>> sortedTotalCommonDF = commonWordsDF.withColumn('totalCount', commonWordsDF.uCount + commonWordsDF.bCount).orderBy("totalCount", ascending=False)
>>> sortedTotalCommonDF.show(20)

```
+--------+------+------+----------+
|    word|uCount|bCount|totalCount|
+--------+------+------+----------+
|    unto|    15|  8997|      9012|
|    lord|   447|  7830|      8277|
|    thou|   161|  5474|      5635|
|     god|   654|  4443|      5097|
|     thy|   141|  4600|      4741|
|      ye|    45|  3982|      4027|
|    thee|    93|  3826|      3919|
|     son|   329|  2370|      2699|
|  israel|    24|  2565|      2589|
|   house|   511|  2024|      2535|
|     day|   751|  1734|      2485|
|    king|   197|  2263|      2460|
|    hand|   918|  1466|      2384|
|  people|   234|  2142|      2376|
|    hath|    39|  2264|      2303|
|    land|   249|  1718|      1967|
|children|   159|  1802|      1961|
|  father|   831|   979|      1810|
|    time|  1141|   623|      1764|
|   shalt|     3|  1616|      1619|
+--------+------+------+----------+
only showing top 20 rows
```

→ 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.

>>> commonWordsDF.orderBy("uCount", ascending = False).show(20)
>>> commonWordsDF.orderBy("bCount", ascending = False).show(20)

```
+-------+------+------+
|   word|uCount|bCount|
+-------+------+------+
|stephen|  1511|     7|
|   time|  1141|   623|
|    yes|  1081|     4|
|   eyes|   987|   503|
|   hand|   918|  1466|
| street|   879|    36|
| little|   870|   242|
| father|   831|   979|
|    day|   751|  1734|
|  round|   717|   320|
|  night|   696|   307|
|   head|   666|   364|
|    sir|   657|    12|
|    god|   654|  4443|
|   name|   651|   929|
|   look|   594|   155|
|   life|   583|   451|
|   john|   582|   139|
|  woman|   558|   357|
|   poor|   558|   205|
+-------+------+------+
only showing top 20 rows
```

```
+--------+------+------+
|    word|uCount|bCount|
+--------+------+------+
|    unto|    15|  8997|
|    lord|   447|  7830|
|    thou|   161|  5474|
|     thy|   141|  4600|
|     god|   654|  4443|
|      ye|    45|  3982|
|    thee|    93|  3826|
|  israel|    24|  2565|
|     son|   329|  2370|
|    hath|    39|  2264|
|    king|   197|  2263|
|  people|   234|  2142|
|   house|   511|  2024|
|children|   159|  1802|
|     day|   751|  1734|
|    land|   249|  1718|
|   shalt|     3|  1616|
|    hand|   918|  1466|
|  saying|   171|  1445|
|  behold|    24|  1326|
+--------+------+------+
only showing top 20 rows
```

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

→ List a random sample containing 5% of words in the final RDD

```
>>> commonWordsDF.count()
5756
```
We need a sample of 5% of words so 288 words

```
>>> randomSample = commonWordsDF.sample(False,0.05)
>>> randomSample.count()
312
>>> randomSample = commonWordsDF.sample(False,0.05)
>>> randomSample.count()
274
```

Close enough. Same explanation as problem 1.

```
>>> randomSample.show()
```

```
+--------+------+------+
|    word|uCount|bCount|
+--------+------+------+
| stephen|  1511|     7|
|   voice|   531|   505|
| forward|   227|    47|
| private|   222|     1|
|    held|   183|    52|
|   speak|   168|   513|
|    wall|   165|   179|
|   horse|   162|    43|
|    boys|   162|     2|
|followed|   161|   108|
|    fell|   156|   243|
|    milk|   153|    48|
|question|   129|    14|
|   wrote|   126|    62|
|    none|   126|   358|
|  beyond|   114|    54|
|   grave|   108|    67|
|  sister|   105|   109|
|carriage|   102|     3|
|   loves|   101|     3|
+--------+------+------+
only showing top 20 rows
```

Assignment 4
CSCI E 63 - Big Data Analytics

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.

Answer:

→ Bring those data in Spark and organize it as DataFrames with named columns.

Read the data from the text files

```
>>> transactionsRDD = sc.textFile("transactions.txt")
```

Separate the lines on the # delimiter

```
>>> transColumns = transactionsRDD.map(lambda t: t.split("#"))
```

Create a dataframe with named columns

```
>>> from pyspark.sql import Row
>>> transactions = transColumns.map(lambda e: Row(date = e[0], time = e[1], customerId = int(e[2]), productId = int(e[3]), qtyBought = int(e[4]), pricePaid = float(e[5])))
>>> transactionsDF = spark.createDataFrame(transactions)
>>> transactionsDF.show()
```

```
+----------+----------+---------+---------+---------+--------+
|customerId|      date|pricePaid|productId|qtyBought|    time|
+----------+----------+---------+---------+---------+--------+
|        51|2015-03-30|  9506.21|       68|        1| 6:55 AM|
|        99|2015-03-30|  4107.59|       86|        5| 7:39 PM|
|        79|2015-03-30|  2987.22|       58|        7|11:57 AM|
|        51|2015-03-30|  7501.89|       50|        6|12:46 AM|
|        86|2015-03-30|   8370.2|       24|        5|11:39 AM|
|        63|2015-03-30|  1023.57|       19|        5|10:35 AM|
|        23|2015-03-30|  5892.41|       77|        7| 2:30 AM|
|        49|2015-03-30|  9298.18|       58|        4| 7:41 PM|
|        97|2015-03-30|  9462.89|       86|        8| 9:18 AM|
|        94|2015-03-30|  4199.15|       26|        4|10:06 PM|
|        91|2015-03-30|  3795.73|       18|        1|10:57 AM|
|        20|2015-03-30|  1477.35|       86|       10| 7:43 AM|
|        38|2015-03-30|   1090.0|       39|        6| 5:58 PM|
|        46|2015-03-30|  1014.78|        6|       10| 1:08 PM|
|        56|2015-03-30|  8346.42|       48|        9|12:18 AM|
|        11|2015-03-30|   364.59|       58|        4| 1:18 AM|
|        59|2015-03-30|  5984.68|        9|        5| 3:01 AM|
|         8|2015-03-30|   1859.2|       35|        6|11:44 AM|
|        23|2015-03-30|  1527.04|        8|        3|12:05 PM|
|        85|2015-03-30|  3314.71|       93|        9| 4:10 AM|
+----------+----------+---------+---------+---------+--------+
only showing top 20 rows
```

Assignment 4
CSCI E 63 - Big Data Analytics

Repeat the same for Products

```
>>> productsRDD = sc.textFile("products.txt")
>>> productsColumns = productsRDD.map(lambda t: t.split("#"))
>>> products = productsColumns.map(lambda e: Row(productId = int(e[0]), productName = e[1],
unitPrice = float(e[2]), qtyAvailable = int(e[3])))
>>> productsDF = spark.createDataFrame(products)
>>> productsDF.show()
```

```
+---------+--------------------+------------+---------+
|productId|         productName|qtyAvailable|unitPrice|
+---------+--------------------+------------+---------+
|        1|ROBITUSSIN PEAK C...|          10|  9721.89|
|        2|Mattel Little Mom...|           6|  6060.78|
|        3|Cute baby doll, b...|           2|  1808.79|
|        4|           Bear doll|           6|    51.06|
|        5|LEGO Legends of C...|           6|   849.36|
|        6|          LEGO Castle|          10|  4777.51|
|        7|          LEGO Mixels|           1|  8720.91|
|        8|       LEGO Star Wars|           4|  7592.44|
|        9|LEGO Lord of the ...|           2|   851.67|
|       10|      LEGO The Hobbit|           9|  7314.55|
|       11|       LEGO Minecraft|           3|  5646.81|
|       12|    LEGO Hero Factory|           1|   6911.2|
|       13|    LEGO Architecture|           5|   604.58|
|       14|         LEGO Technic|           3|  7423.48|
|       15|LEGO Storage & Ac...|           2|  3125.96|
|       16|         LEGO Classic|          10|   9933.3|
|       17|    LEGO Galaxy Squad|           4|  5593.16|
|       18|      LEGO Mindstorms|          10|  6022.88|
|       19|     LEGO Minifigures|           1|  5775.99|
|       20|           LEGO Elves|           4|  4589.79|
+---------+--------------------+------------+---------+
only showing top 20 rows
```

→ Find 5 customers with the largest spent on the day.

Create a temp view to run regular SQL query

```
>>> transactionsDF.createOrReplaceTempView("transactionVW")
>>> productsDF.createOrReplaceTempView("productsVW")
```

Run a SQL query on the view to retrieve top 5 spending customers

```
>>> top5Customers = spark.sql("SELECT customerId FROM (SELECT customerId, SUM(pricePaid)
AS totalPrice FROM transactionVW GROUP BY customerId ORDER BY totalPrice desc LIMIT 5)")
>>> top5Customers.show()
```

```
+----------+
|customerId|
+----------+
|        76|
|        53|
|        56|
|        51|
|        31|
+----------+
```

Assignment 4
CSCI E 63 - Big Data Analytics

→ Find the names of the products each of those 5 customers bought.

Create a view of the top 5 customers to query against it

>>> top5Customers.createOrReplaceTempView("top5CustomersVW")

>>> top5CustomersProducts = spark.sql("SELECT distinct(productName) from productsVW prd, transactionVW tran WHERE prd.productId = tran.productId and tran.customerId in(SELECT customerId FROM top5CustomersVW)")

>>> top5CustomersProducts.show()

```
+--------------------+
|         productName|
+--------------------+
|ATOPALM MUSCLE AN...|
|Far Cry 4 Limited...|
|healthy accents s...|
|Brimonidine Tartrate|
|Treatment Set TS3...|
|Star Wars Republi...|
|AMBROSIA TRIFIDA ...|
|Essentials Dantes...|
|            Acyclovir|
|               Grippe|
|     LEGO The Hobbit|
|Essentials Medal ...|
|SAMSUNG LED TV 39...|
|Notebook Lenovo E...|
|PC HP 490PD MT, D...|
|    chest congestion|
|Notebook Lenovo U...|
| LEGO Speed Champion|
|PC HP 600PD TWR, ...|
|                Jafra|
+--------------------+
only showing top 20 rows
```

→ Find the names and total number sold of 10 most popular products.

>>> top10PopularProductNames = spark.sql("SELECT prd.productName, SUM(qtyBought) AS totalQty FROM transactionVW tran, productsVW prd WHERE prd.productId = tran.productId GROUP BY prd.productName ORDER BY totalQty desc LIMIT 10")

>>> top10PopularProductNames.show()

```
+--------------------+--------+
|         productName|totalQty|
+--------------------+--------+
|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|
+--------------------+--------+
```

Assignment 4
CSCI E 63 - Big Data Analytics

→ Order products per the number sold.

>>> productsOrderedPerSold = spark.sql("SELECT productId,productName,qtyAvailable,unitPrice FROM (SELECT prd.productId,prd.productName,prd.qtyAvailable, prd.unitPrice, SUM(qtyBought) AS totalQty FROM transactionVW tran, productsVW prd WHERE prd.productId = tran.productId GROUP BY prd.productId,prd.productName,prd.qtyAvailable, prd.unitPrice ORDER BY totalQty desc)")

>>> productsOrderedPerSold.show()

```
+---------+--------------------+------------+---------+
|productId|         productName|qtyAvailable|unitPrice|
+---------+--------------------+------------+---------+
|       58|Notebook Lenovo U...|           3|   461.08|
|       44|SAMSUNG LED TV 39...|           1|  2531.15|
|       86|            Jantoven|           9|   3255.4|
|       93|               Jafra|           4|  3715.07|
|       28|Far Cry 4 Limited...|           1|   711.88|
|       65|Roller Derby Roll...|           5|  7783.79|
|       30|Procesor Intel Co...|           6|  4570.99|
|       38|  Sony Playstation 3|           4|  5088.35|
|       96|    chest congestion|           1|  1305.04|
|       26|Barbie Beach Ken ...|           5|   742.84|
|        4|           Bear doll|           6|    51.06|
|       50|  LG LED TV 32LN575S|           6|  8379.93|
|       37|GAM X360 Need for...|           8|  6790.22|
|        7|         LEGO Mixels|           1|  8720.91|
|       57|Notebook Lenovo U...|           2|  2626.88|
|        6|         LEGO Castle|          10|  4777.51|
|       62|PC HP 490PD MT, D...|           3|  6248.36|
|       61|PC HP 490PD MT, D...|           4|  3906.32|
|      100|               ZOCOR|           8|  7040.56|
|        1|ROBITUSSIN PEAK C...|          10|  9721.89|
+---------+--------------------+------------+---------+
only showing top 20 rows
```

Assignment 4
CSCI E 63 - Big Data Analytics

→ Order products by the total value (quanity*price) sold.

>>> productsOrderedByValue = spark.sql("SELECT productId,productName,qtyAvailable,unitPrice FROM (SELECT prd.productId,prd.productName,prd.qtyAvailable, prd.unitPrice, SUM(tran.qtyBought * prd.unitPrice) AS totalValue FROM transactionVW tran, productsVW prd WHERE prd.productId = tran.productId GROUP BY prd.productId,prd.productName,prd.qtyAvailable, prd.unitPrice ORDER BY totalValue desc)")

>>> productsOrderedByValue.show()

```
+---------+--------------------+------------+---------+
|productId|         productName|qtyAvailable|unitPrice|
+---------+--------------------+------------+---------+
|        1|ROBITUSSIN PEAK C...|          10|  9721.89|
|       65|Roller Derby Roll...|           5|  7783.79|
|        7|         LEGO Mixels|           1|  8720.91|
|       50|  LG LED TV 32LN575S|           6|  8379.93|
|       98|          Gabapentin|           5|  8763.57|
|       22| LEGO Speed Champion|           2|  8486.42|
|       16|        LEGO Classic|          10|   9933.3|
|       69|           ibuprofen|           4|  7907.21|
|       37|GAM X360 Need for...|           8|  6790.22|
|       89|           Glipizide|           5|  9376.44|
|      100|               ZOCOR|           8|  7040.56|
|       46|SAMSUNG LED TV 32...|           1|  8508.89|
|       51|Essentials Tekken...|           7|   8875.2|
|       62|PC HP 490PD MT, D...|           3|  6248.36|
|       59|PC HP 600PD TWR, ...|           1|   6326.7|
|       47|SAMSUNG LED TV 55...|           9|  7673.37|
|       97|Santalia Clinical...|           1|  8835.52|
|       48|  LG LED TV 42LA6130|           8|  6918.75|
|       41|Star Wars Republi...|           2|   8673.6|
|       38|  Sony Playstation 3|           4|  5088.35|
+---------+--------------------+------------+---------+
only showing top 20 rows
```

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

Problem 4.
Implement problem 3 using RDD APIs.

Answer:

→ Bring those data in Spark as RDD

Read the data from the text files

```
>>> transactionsRDD = sc.textFile("transactions.txt")
>>> productsRDD = sc.textFile("products.txt")
```

Separate the lines on the # delimiter

```
>>> transColumns = transactionsRDD.map(lambda t: t.split("#"))
productsColumns = productsRDD.map(lambda t: t.split("#"))
```

Verify the separated RDDs

```
>>> transColumns.take(10)
[[u'2015-03-30', u'6:55 AM', u'51', u'68', u'1', u'9506.21'], [u'2015-03-30', u'7:39 PM', u'99', u'86', u'5',
u'4107.59'], [u'2015-03-30', u'11:57 AM', u'79', u'58', u'7', u'2987.22'], [u'2015-03-30', u'12:46 AM',
u'51', u'50', u'6', u'7501.89'], [u'2015-03-30', u'11:39 AM', u'86', u'24', u'5', u'8370.2'], [u'2015-03-30',
u'10:35 AM', u'63', u'19', u'5', u'1023.57'], [u'2015-03-30', u'2:30 AM', u'23', u'77', u'7', u'5892.41'],
[u'2015-03-30', u'7:41 PM', u'49', u'58', u'4', u'9298.18'], [u'2015-03-30', u'9:18 AM', u'97', u'86', u'8',
u'9462.89'], [u'2015-03-30', u'10:06 PM', u'94', u'26', u'4', u'4199.15']]
>>>productsColumns.take(10)
[[u'1', u'ROBITUSSIN PEAK COLD NIGHTTIME COLD PLUS FLU', u'9721.89', u'10'], [u'2',
u'Mattel Little Mommy Doctor Doll', u'6060.78', u'6'], [u'3', u'Cute baby doll, battery', u'1808.79', u'2'],
[u'4', u'Bear doll', u'51.06', u'6'], [u'5', u'LEGO Legends of Chima', u'849.36', u'6'], [u'6', u'LEGO Castle',
u'4777.51', u'10'], [u'7', u'LEGO Mixels', u'8720.91', u'1'], [u'8', u'LEGO Star Wars', u'7592.44', u'4'],
[u'9', u'LEGO Lord of the Rings', u'851.67', u'2'], [u'10', u'LEGO The Hobbit', u'7314.55', u'9']]
```

→ Find 5 customers with the largest spent on the day.

Create RDD with customer IDs and Amount spent. We need the amount spent column to be a number.

```
>>> customerAmounts = transColumns.map(lambda x: (x[2],float(x[5])))
>>> customerAmounts.take(10)
[(u'51', 9506.21), (u'99', 4107.59), (u'79', 2987.22), (u'51', 7501.89), (u'86', 8370.2), (u'63', 1023.57),
(u'23', 5892.41), (u'49', 9298.18), (u'97', 9462.89), (u'94', 4199.15)]
```

Assignment 4
CSCI E 63 - Big Data Analytics

Sum up the amounts to get total amounts for each customer

```
>>> customerTotalAmounts = customerAmounts.reduceByKey(lambda a,b:a+b)
>>> customerTotalAmounts.take(10)
[(u'24', 39375.28), (u'25', 62861.79999999999), (u'26', 74109.66), (u'27', 57023.96), (u'20',
32997.799999999996), (u'21', 62274.25), (u'22', 43987.57), (u'23', 62269.11000000001), (u'28',
45534.3), (u'29', 31389.32)]
```

Sort and get the top 5 spending customers

```
>>> sortedCustomerTotals = customerTotalAmounts.sortBy(lambda (a,b): b, ascending = False)
>>> sortedCustomerTotals.take(5)
[(u'76', 100049.00000000001), (u'53', 88829.76000000001), (u'56', 85906.94), (u'51', 83312.12), (u'31',
83202.61)]
```

Assign them to a RDD

```
>>> top5CustomerTotals = sortedCustomerTotals.take(5)
>>> print(top5CustomerTotals)
[(u'76', 100049.00000000001), (u'53', 88829.76000000001), (u'56', 85906.94), (u'51', 83312.12), (u'31',
83202.61)]
>>> top5CustomersRDD = sc.parallelize(top5CustomerTotals)
>>> top5CustomersRDD.take(5)
[(u'76', 100049.00000000001), (u'53', 88829.76000000001), (u'56', 85906.94), (u'51', 83312.12), (u'31',
83202.61)]
```

→ Find the names of the products each of those 5 customers bought.

Create RDD of the 5 customers

```
>>> top5CustomersIds = top5CustomersRDD.map(lambda x : x[0])
>>> top5CustomersIds.take(5)
[u'76', u'53', u'56', u'51', u'31']
```

Get the list of transactions for these customer IDs

```
>>> top5CustomersTransactions = transColumns.filter(lambda x: x[2] in [’76’,'56','53','51','31'])
>>> top5CustomersTransactions.take(10)
[[u'2015-03-30', u'6:55 AM', u'51', u'68', u'1', u'9506.21'], [u'2015-03-30', u'12:46 AM', u'51', u'50', u'6',
u'7501.89'], [u'2015-03-30', u'12:18 AM', u'56', u'48', u'9', u'8346.42'], [u'2015-03-30', u'10:18 AM',
u'51', u'44', u'4', u'5231.69'], [u'2015-03-30', u'6:18 AM', u'53', u'42', u'5', u'2197.85'], [u'2015-03-30',
u'7:39 AM', u'51', u'77', u'3', u'4937.79'], [u'2015-03-30', u'5:47 PM', u'51', u'1', u'8', u'9086.1'],
[u'2015-03-30', u'1:56 AM', u'56', u'28', u'5', u'2387.26'], [u'2015-03-30', u'3:40 PM', u'76', u'12', u'5',
u'8706.91'], [u'2015-03-30', u'11:48 PM', u'56', u'62', u'7', u'9248.15']]
```

Assignment 4
CSCI E 63 - Big Data Analytics

Get the list of products for these customer IDs

```
>>> top5CustomersProductIds = top5CustomersTransactions.map(lambda x: (x[2],x[3]))
>>> top5CustomersProductIds.take(10)
[(u'68', u'51'), (u'50', u'51'), (u'48', u'56'), (u'44', u'51'), (u'42', u'53'), (u'77', u'51'), (u'1', u'51'), (u'28', u'56'), (u'12', u'76'), (u'62', u'56')]
```

Create a product guide

```
>>> productGuide = productsColumns.map(lambda x : (x[0],x[1]))
>>> productGuide.take(10)
[(u'1', u'ROBITUSSIN PEAK COLD NIGHTTIME COLD PLUS FLU'), (u'2', u'Mattel Little Mommy Doctor Doll'), (u'3', u'Cute baby doll, battery'), (u'4', u'Bear doll'), (u'5', u'LEGO Legends of Chima'), (u'6', u'LEGO Castle'), (u'7', u'LEGO Mixels'), (u'8', u'LEGO Star Wars'), (u'9', u'LEGO Lord of the Rings'), (u'10', u'LEGO The Hobbit')]
```

Join the two to get an RDD with key as the product ID and value of customer ID, product Name

```
>>> top5CustomersProducts = top5CustomersProductIds.join(productGuide)
>>> top5CustomersProducts.take(10)
[(u'42', (u'53', u'Star Wars The Force Unleashed Ultimate Sith Edition PC')), (u'82', (u'31', u'Scrub Care Povidone Iodine Cleansing Scrub')), (u'62', (u'56', u'PC HP 490PD MT, D5T60EA')), (u'62', (u'31', u'PC HP 490PD MT, D5T60EA')), (u'66', (u'31', u'Stomach Disorders')), (u'68', (u'51', u'Niacin')), (u'68', (u'53', u'Niacin')), (u'93', (u'76', u'Jafra')), (u'93', (u'53', u'Jafra')), (u'93', (u'56', u'Jafra'))]
```

Get just the product names

```
>>> top5CustomersProductNames = top5CustomersProducts.map(lambda (x,(a,b)): b)
>>> top5CustomersProductNames.take(10)
[u'Star Wars The Force Unleashed Ultimate Sith Edition PC', u'Scrub Care Povidone Iodine Cleansing Scrub', u'PC HP 490PD MT, D5T60EA', u'PC HP 490PD MT, D5T60EA', u'Stomach Disorders', u'Niacin', u'Niacin', u'Jafra', u'Jafra', u'Jafra']
```

Get the distinct product names

```
>>> top5CustomersDistinctProductNames = top5CustomersProductNames.distinct()
>>> top5CustomersDistinctProductNames.take(10)
[u'Niacin', u'GUNA-EGF', u'Glipizide', u'Treatment Set TS332287', u'Star Wars Republic Commando PC', u'LEGO Castle', u'Notebook Lenovo U330p, 59-390439', u'Barbie Beach Ken Doll', u'LEGO Hero Factory', u'SAMSUNG LED TV 55F6500, Full HD, 3D, USB']
```

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

→ Find the names and total number sold of 10 most popular products.

Get a count of the products sold and sort in descending order

```
>>> soldProductCount = transColumns.map(lambda x: (x[3],1)).reduceByKey(lambda a, b: a +
b).sortBy(lambda (a,b):b, ascending=False)
>>> soldProductCount.take(10)
[(u'58', 39), (u'44', 25), (u'28', 19), (u'93', 19), (u'86', 17), (u'59', 16), (u'57', 16), (u'30', 16), (u'1', 16),
(u'4', 15)]
```

Add the product name for details and sort by count

```
>>> soldProductDetailCount = soldProductCount.join(productGuide)
>>> sortedSoldProductDetailCount = soldProductDetailCount.sortBy(lambda (x,
(a,b)):a,ascending=False)
>>> sortedSoldProductDetailCount.take(10)
[(u'58', (39, u'Notebook Lenovo U330p, 59-390439')), (u'44', (25, u'SAMSUNG LED TV 39F5500, Full
HD, USB')), (u'93', (19, u'Jafra')), (u'28', (19, u'Far Cry 4 Limited Edition for Xbox One')), (u'86', (17,
u'Jantoven')), (u'1', (16, u'ROBITUSSIN PEAK COLD NIGHTTIME COLD PLUS FLU')), (u'59', (16,
u'PC HP 600PD TWR, E7P49AW')), (u'57', (16, u'Notebook Lenovo U430p, 59-390459')), (u'30', (16,
u'Procesor Intel Core i5 3470')), (u'7', (15, u'LEGO Mixels'))]
```

→ Order products per the number sold.

Convert the products to a key value pair

```
>>> productsColumns = productsColumns.map(lambda (a,b,c,d):(a,(b,c,d)))
```

Join the RDD soldProductCount built earlier to the products to add a count column and sort in descending order of the count column

```
>>> productsColumnsWithCount = productsColumns.join(soldProductCount).sortBy(lambda (x,
((a,b,c),d)): d, ascending = False)

>>> productsColumnsWithCount.take(10)
[(u'58', ((u'Notebook Lenovo U330p, 59-390439', u'461.08', u'3'), 39)), (u'44', ((u'SAMSUNG LED TV
39F5500, Full HD, USB', u'2531.15', u'1'), 25)), (u'93', ((u'Jafra', u'3715.07', u'4'), 19)), (u'28', ((u'Far
Cry 4 Limited Edition for Xbox One', u'711.88', u'1'), 19)), (u'86', ((u'Jantoven', u'3255.4', u'9'), 17)),
(u'1', ((u'ROBITUSSIN PEAK COLD NIGHTTIME COLD PLUS FLU', u'9721.89', u'10'), 16)), (u'59',
((u'PC HP 600PD TWR, E7P49AW', u'6326.7', u'1'), 16)), (u'57', ((u'Notebook Lenovo U430p,
59-390459', u'2626.88', u'2'), 16)), (u'30', ((u'Procesor Intel Core i5 3470', u'4570.99', u'6'), 16)), (u'7',
((u'LEGO Mixels', u'8720.91', u'1'), 15))]
```

Karan A. Bhandarkar

Assignment 4
CSCI E 63 - Big Data Analytics

→ Order products by the total value (quanity*price) sold.

Join the RDD soldProductCount built earlier to the products to add a count column and sort in descending order of the total value i.e. quantity * price i.e. ‘d’*‘b’

```
>>> productsColumnsWithCount = productsColumns.join(soldProductCount).sortBy(lambda (x, ((a,b,c),d)): d*b, ascending = False)

>>> productsColumnsWithCount.take(10)
[(u'16', ((u'LEGO Classic', u'9933.3', u'10'), 8)), (u'33', ((u'Intel Core i5 3330', u'9785.44', u'9'), 4)), (u'1', ((u'ROBITUSSIN PEAK COLD NIGHTTIME COLD PLUS FLU', u'9721.89', u'10'), 16)), (u'83', ((u'Ativan', u'9511.99', u'9'), 5)), (u'89', ((u'Glipizide', u'9376.44', u'5'), 12)), (u'51', ((u'Essentials Tekken 6 PS3', u'8875.2', u'7'), 11)), (u'97', ((u'Santalia Clinical Intenstive Spot Treatment', u'8835.52', u'1'), 9)), (u'98', ((u'Gabapentin', u'8763.57', u'5'), 12)), (u'7', ((u'LEGO Mixels', u'8720.91', u'1'), 15)), (u'72', ((u'Obao', u'8693.64', u'8'), 7))]
```

Karan A. Bhandarkar