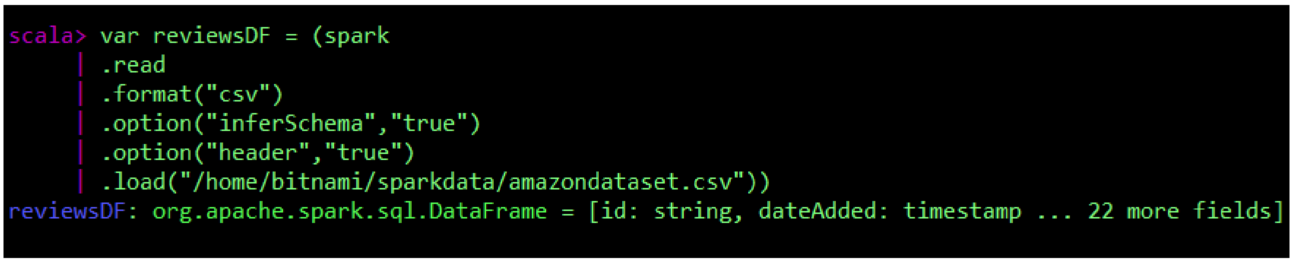
Big Data Hadoop Project
(Amazon Reviews—Sentiment Data)

1. Analyze all the positive reviews (Any review with a rating of 4 and above is considered positive) and find out the top 20 words used in those positive reviews.

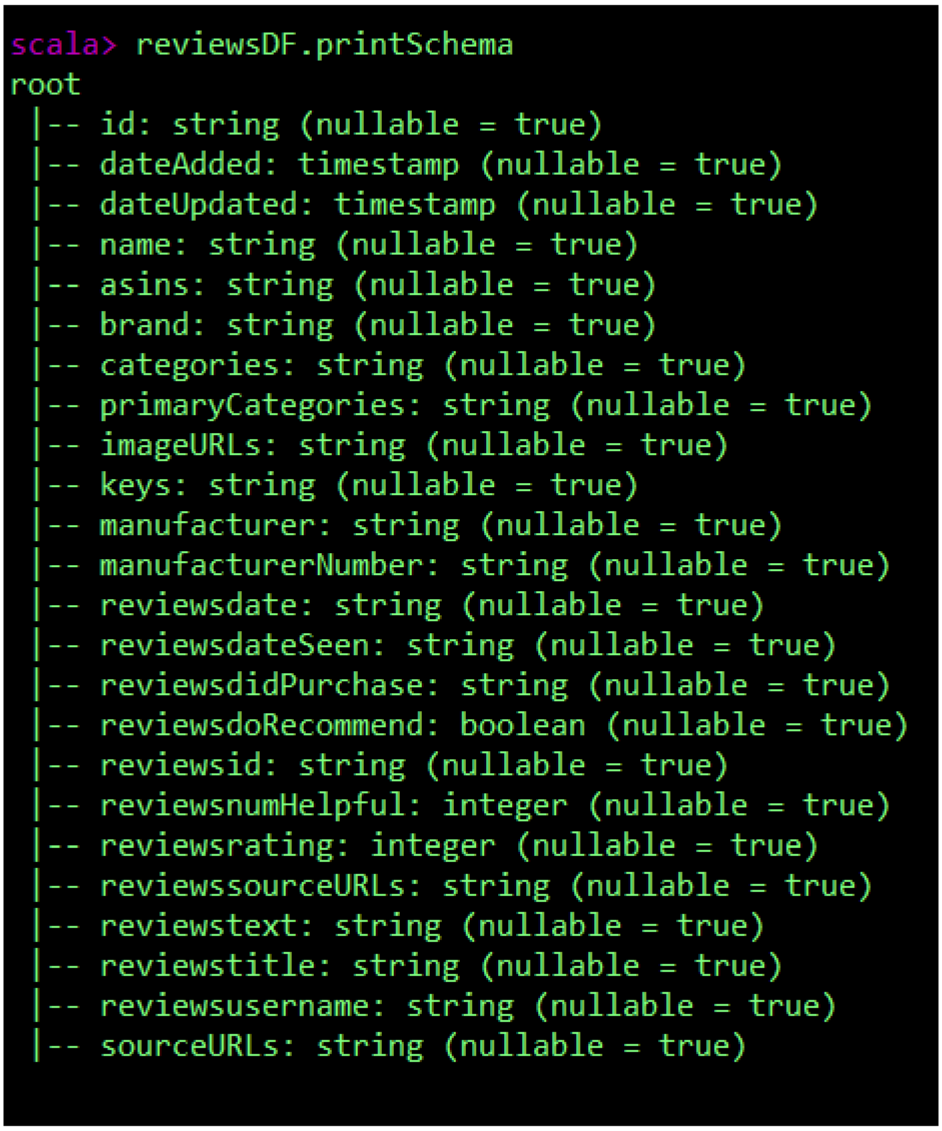
Solution:

Loading the reviews data



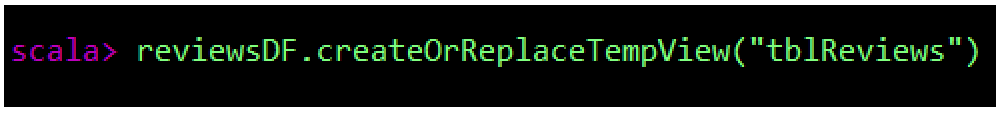
```
scala> var reviewsDF = (spark
     | .read
     | .format("csv")
     | .option("inferSchema","true")
     | .option("header","true")
     | .load("/home/bitnami/sparkdata/amazondataset.csv"))
reviewsDF: org.apache.spark.sql.DataFrame = [id: string, dateAdded: timestamp ... 22 more fields]
```

Viewing Schema



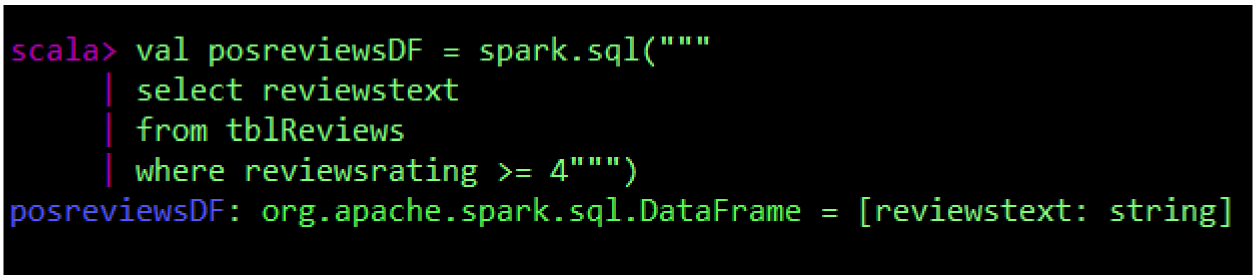
```
scala> reviewsDF.printSchema
root
 |-- id: string (nullable = true)
 |-- dateAdded: timestamp (nullable = true)
 |-- dateUpdated: timestamp (nullable = true)
 |-- name: string (nullable = true)
 |-- asins: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- primaryCategories: string (nullable = true)
 |-- imageURLs: string (nullable = true)
 |-- keys: string (nullable = true)
 |-- manufacturer: string (nullable = true)
 |-- manufacturerNumber: string (nullable = true)
 |-- reviewsdate: string (nullable = true)
 |-- reviewsdateSeen: string (nullable = true)
 |-- reviewsdidPurchase: string (nullable = true)
 |-- reviewsdoRecommend: boolean (nullable = true)
 |-- reviewsid: string (nullable = true)
 |-- reviewsnumHelpful: integer (nullable = true)
 |-- reviewsrating: integer (nullable = true)
 |-- reviewssourceURLs: string (nullable = true)
 |-- reviewstext: string (nullable = true)
 |-- reviewstitle: string (nullable = true)
 |-- reviewsusername: string (nullable = true)
 |-- sourceURLs: string (nullable = true)
```

Creating a View



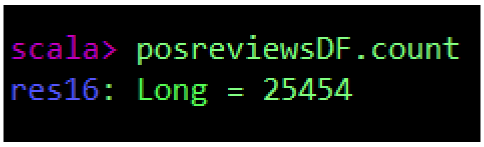
```
scala> reviewsDF.createOrReplaceTempView("tblReviews")
```

Selecting Positive Reviews (reviwsrating >= 4)



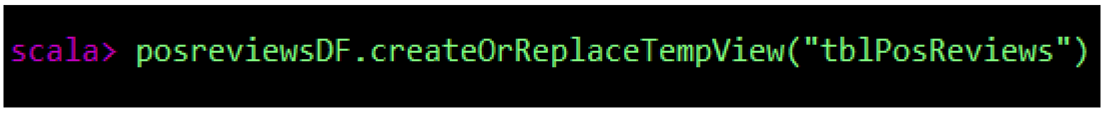
```
scala> val posreviewsDF = spark.sql("""
     | select reviewstext
     | from tblReviews
     | where reviewsrating >= 4""")
posreviewsDF: org.apache.spark.sql.DataFrame = [reviewstext: string]
```

Checking total count of positive reviews



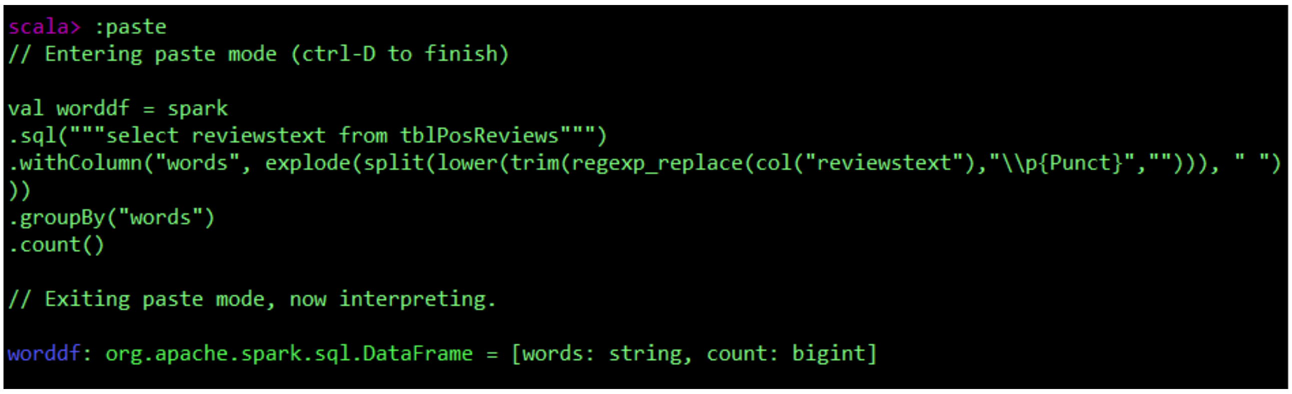
```
scala> posreviewsDF.count
res16: Long = 25454
```

Creating a View of Positive Reviews



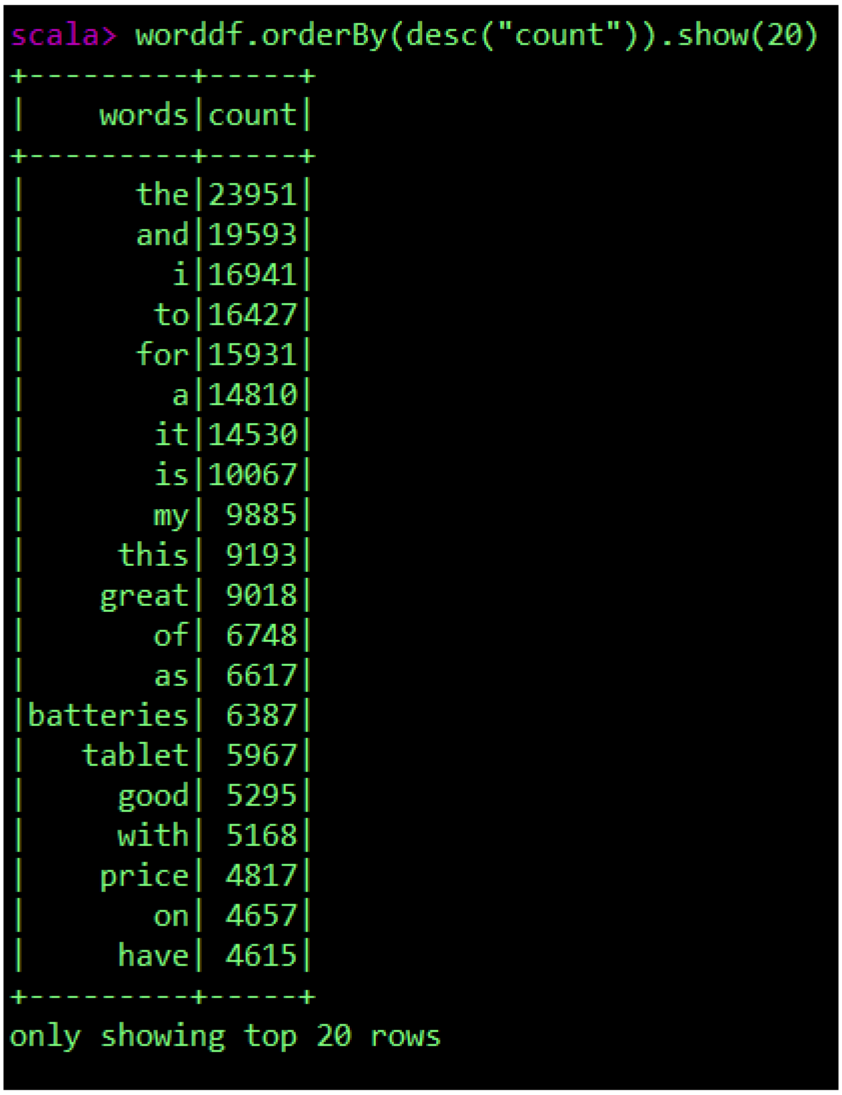
```
scala> posreviewsDF.createOrReplaceTempView("tblPosReviews")
```

Wordcount Program



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val worddf = spark
.sql("""select reviewstext from tblPosReviews""")
.withColumn("words", explode(split(lower(trim(regexp_replace(col("reviewstext"),"\\p{Punct}",""))), " ")
))
.groupBy("words")
.count()

// Exiting paste mode, now interpreting.

worddf: org.apache.spark.sql.DataFrame = [words: string, count: bigint]
```

Generating output for top 20 words by count in positive reviews

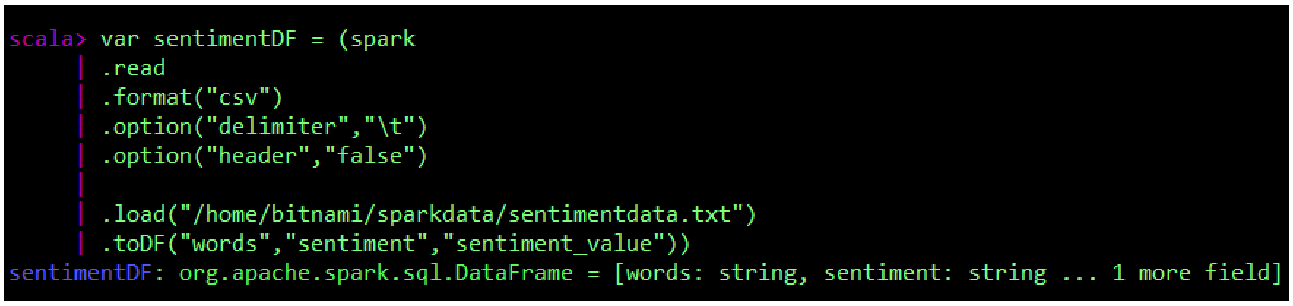


```
scala> worddf.orderBy(desc("count")).show(20)
+---------+-----+
|    words|count|
+---------+-----+
|      the|23951|
|      and|19593|
|        i|16941|
|       to|16427|
|      for|15931|
|        a|14810|
|       it|14530|
|       is|10067|
|       my| 9885|
|     this| 9193|
|    great| 9018|
|       of| 6748|
|       as| 6617|
|batteries| 6387|
|   tablet| 5967|
|     good| 5295|
|     with| 5168|
|    price| 4817|
|       on| 4657|
|     have| 4615|
+---------+-----+
only showing top 20 rows
```

2. Use the word sentiment dataset and find out the percentage of words that are positive, negative and neutral. The words that aren't mentioned in the word sentiment dataset are considered as neutral.

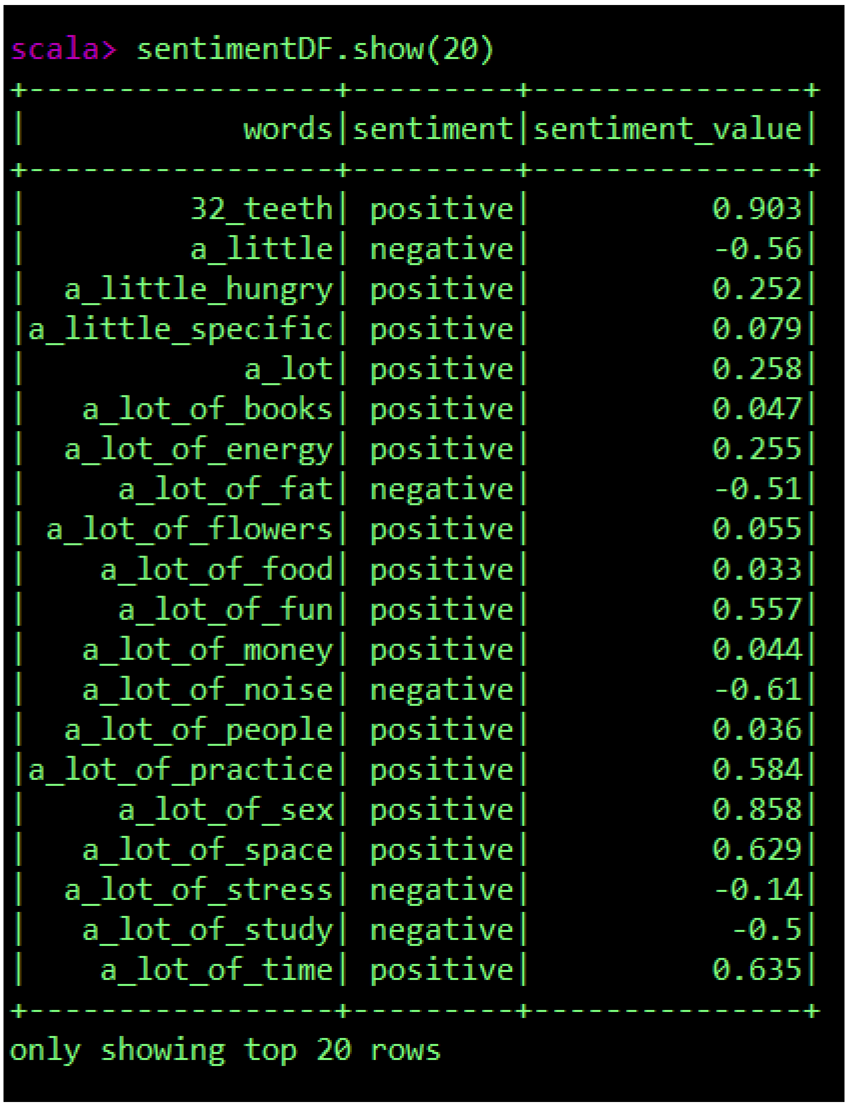
Solution:

Loading Word Sentiment data



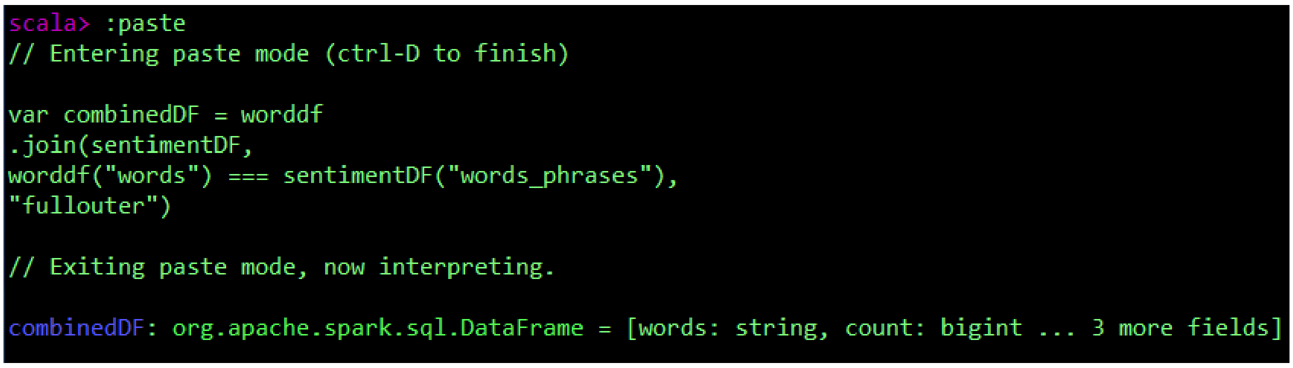
```
scala> var sentimentDF = (spark
     | .read
     | .format("csv")
     | .option("delimiter","\t")
     | .option("header","false")
     |
     | .load("/home/bitnami/sparkdata/sentimentdata.txt")
     | .toDF("words","sentiment","sentiment_value"))
sentimentDF: org.apache.spark.sql.DataFrame = [words: string, sentiment: string ... 1 more field]
```

Viewing first 20 records



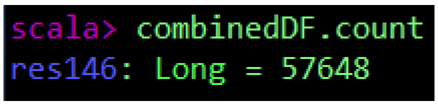
```
scala> sentimentDF.show(20)
+-----------------+---------+---------------+
|            words|sentiment|sentiment_value|
+-----------------+---------+---------------+
|         32_teeth| positive|          0.903|
|         a_little| negative|          -0.56|
|  a_little_hungry| positive|          0.252|
|a_little_specific| positive|          0.079|
|            a_lot| positive|          0.258|
|   a_lot_of_books| positive|          0.047|
|  a_lot_of_energy| positive|          0.255|
|     a_lot_of_fat| negative|          -0.51|
| a_lot_of_flowers| positive|          0.055|
|    a_lot_of_food| positive|          0.033|
|     a_lot_of_fun| positive|          0.557|
|   a_lot_of_money| positive|          0.044|
|   a_lot_of_noise| negative|          -0.61|
|  a_lot_of_people| positive|          0.036|
|a_lot_of_practice| positive|          0.584|
|     a_lot_of_sex| positive|          0.858|
|   a_lot_of_space| positive|          0.629|
|  a_lot_of_stress| negative|          -0.14|
|   a_lot_of_study| negative|           -0.5|
|    a_lot_of_time| positive|          0.635|
+-----------------+---------+---------------+
only showing top 20 rows
```

Combining Word Sentiment data to find count of pos, neg and neutral



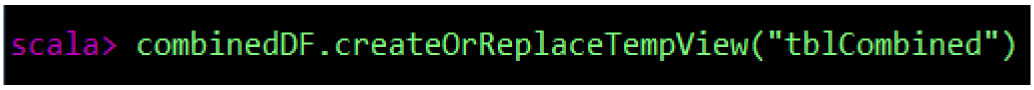
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

var combinedDF = worddf
.join(sentimentDF,
worddf("words") === sentimentDF("words_phrases"),
"fullouter")

// Exiting paste mode, now interpreting.

combinedDF: org.apache.spark.sql.DataFrame = [words: string, count: bigint ... 3 more fields]
```

Viewing count



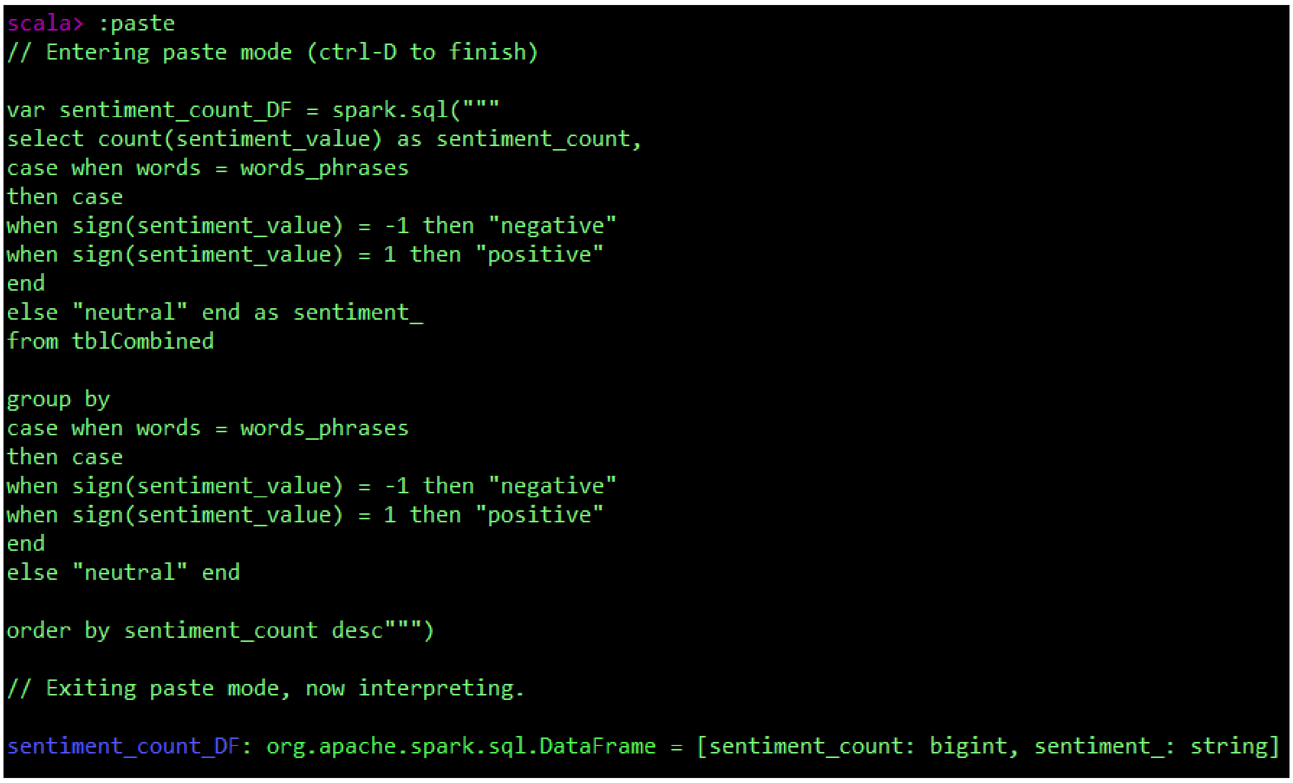
```
scala> combinedDF.count
res146: Long = 57648
```

Creating a View



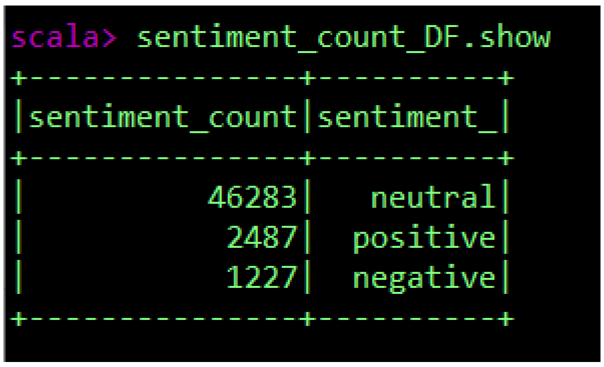
```
scala> combinedDF.createOrReplaceTempView("tblCombined")
```

Creating sentiment count DF (with pos, neg and neutral sentiments)



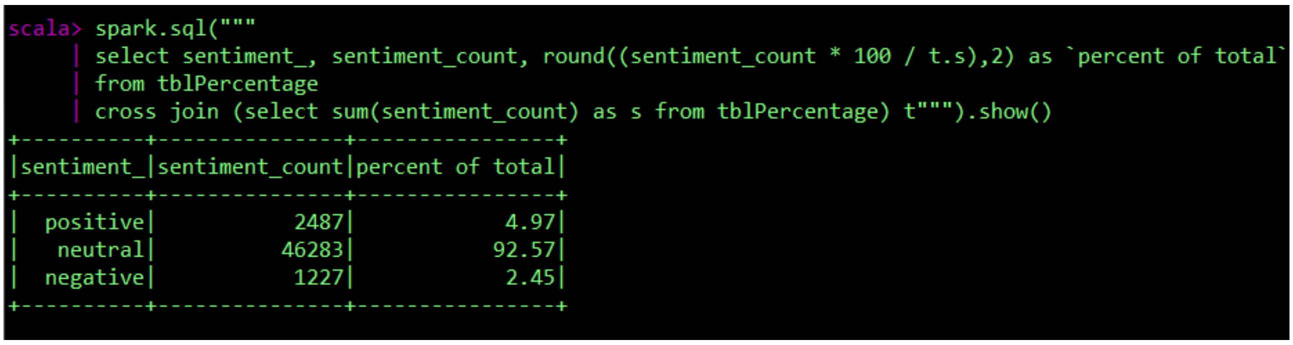
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

var sentiment_count_DF = spark.sql("""
select count(sentiment_value) as sentiment_count,
case when words = words_phrases
then case
when sign(sentiment_value) = -1 then "negative"
when sign(sentiment_value) = 1 then "positive"
end
else "neutral" end as sentiment_
from tblCombined

group by
case when words = words_phrases
then case
when sign(sentiment_value) = -1 then "negative"
when sign(sentiment_value) = 1 then "positive"
end
else "neutral" end

order by sentiment_count desc""")

// Exiting paste mode, now interpreting.

sentiment_count_DF: org.apache.spark.sql.DataFrame = [sentiment_count: bigint, sentiment_: string]
```

Viewing count of positive, negative and neutral words



```
scala> sentiment_count_DF.show
+---------------+----------+
|sentiment_count|sentiment_|
+---------------+----------+
|          46283|   neutral|
|           2487|  positive|
|           1227|  negative|
+---------------+----------+
```

Generating output for percentage of positive, negative and neutral words



```
scala> spark.sql("""
     | select sentiment_, sentiment_count, round((sentiment_count * 100 / t.s),2) as `percent of total`
     | from tblPercentage
     | cross join (select sum(sentiment_count) as s from tblPercentage) t""").show()
+----------+---------------+----------------+
|sentiment_|sentiment_count|percent of total|
+----------+---------------+----------------+
|  positive|           2487|            4.97|
|   neutral|          46283|           92.57|
|  negative|           1227|            2.45|
+----------+---------------+----------------+
```