Assignment 20.2

Step1: Download the demonetization_tweets.csv from the Assignment Link given and copy to /home/acadgild/assignment_20.2

Step2: Download AFINN.txt from the URL below

https://raw.githubusercontent.com/wendykan/twitter-sentiment-analysis/master/AFINN-111.txt

Step3: Process the tweets from demonetization-tweets.csv and store to temporary table tweets

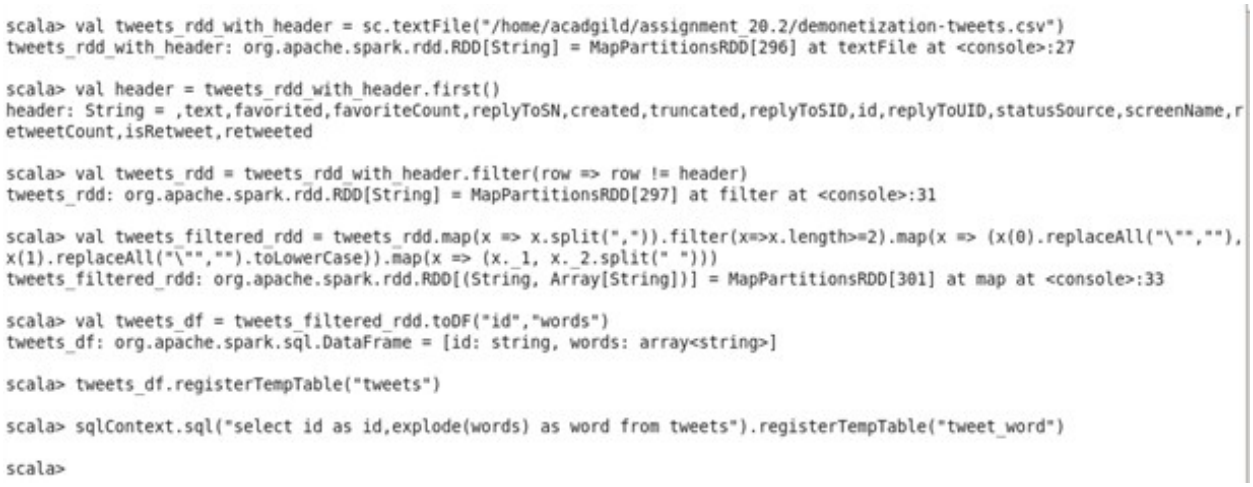
- Create RDD from /home/acadgild/assignment_20.2/demonetization-tweets.csv and put to tweets_rdd_with_header
- Filter the header from tweets_rdd_with_header and assign the RDD to tweets_rdd
- Filter the records based on field separator comma (,) and filter all the records who has numbers of fields is greater than 2 and remove all quote characters, and map only first two fields and put the result to tweets_filtered_rdd
- Create a dataframe with two fields id, words from tweets_filtered_rdd to
- Create a temporary tables tweets
- Explode words and create a table tweet_words

Code is as below:

```
val tweets_rdd_with_header = sc.textFile("/home/acadgild/assignment_20.2/demonetization-tweets.csv")

val header = tweets_rdd_with_header.first()

val tweets_rdd = tweets_rdd_with_header.filter(row => row != header)

val tweets_filtered_rdd = tweets_rdd.map(x => x.split(",")).filter(x=>x.length>=2).map(x => (x(0).replaceAll("\"",""),x(1).replaceAll("\"","").toLowerCase)).map(x => (x._1, x._2.split(" ")))

val tweets_df = tweets_filtered_rdd.toDF("id","words")

tweets_df.registerTempTable("tweets")
```

sqlContext.sql("select id as id,explode(words) as word from tweets").registerTempTable("tweet_word")

Screenshot is as below:



```
scala> val tweets_rdd_with_header = sc.textFile("/home/acadgild/assignment_20.2/demonetization-tweets.csv")
tweets_rdd_with_header: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[296] at textFile at <console>:27

scala> val header = tweets_rdd_with_header.first()
header: String = ,text,favorited,favoriteCount,replyToSN,created,truncated,replyToSID,id,replyToUID,statusSource,screenName,r
etweetCount,isRetweet,retweeted

scala> val tweets_rdd = tweets_rdd_with_header.filter(row => row != header)
tweets_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[297] at filter at <console>:31

scala> val tweets_filtered_rdd = tweets_rdd.map(x => x.split(",")).filter(x=>x.length>=2).map(x => (x(0).replaceAll("\"",""),
x(1).replaceAll("\"","").toLowerCase)).map(x => (x._1, x._2.split(" ")))
tweets_filtered_rdd: org.apache.spark.rdd.RDD[(String, Array[String])] = MapPartitionsRDD[301] at map at <console>:33

scala> val tweets_df = tweets_filtered_rdd.toDF("id","words")
tweets_df: org.apache.spark.sql.DataFrame = [id: string, words: array<string>]

scala> tweets_df.registerTempTable("tweets")

scala> sqlContext.sql("select id as id,explode(words) as word from tweets").registerTempTable("tweet_word")

scala>
```

Step4: Process AFINN.txt and store to temporary table afinn

- Create RDD from ("/home/acadgild/assignment_20.2/AFINN.txt and put to afinn_rdd
- Split words of afinn_rdd based on tab and create a Dataframe afinn_df with fields word, rating
- Create temporary table afinn

val afinn_rdd = sc.textFile("/home/acadgild/assignment_20.2/AFINN.txt")

val afinn_df = afinn_rdd.map(x => x.split("\t")).map(x => (x(0),x(1))).toDF("word","rating")

afinn_df.registerTempTable("afinn")

sqlContext.sql("select t.id,AVG(a.rating) as rating from tweet_word t join afinn a on t.word=a.word group by t.id order by rating desc").show(100)

Step5: Find the the rating for tweet

Join the tables tweet_word and afinn on word and using avg function calculate average of rating group by tweet id, descending order by rating. Showing first 100 records of results

Code is as below:

sqlContext.sql("select t.id,AVG(a.rating) as rating from tweet_word t join afinn a on t.word=a.word group by t.id order by rating desc").show(100)

```
r by    rating desc").show(100)
+----+------+
|  id|rating|
+----+------+
|7025|   4.0|
|7281|   4.0|
|3822|   4.0|
|6546|   4.0|
|6610|   4.0|
|5733|   4.0|
|4185|   4.0|
|7994|   4.0|
| 308|   3.5|
|5876|   3.0|
|6922|   3.0|
|7990|   3.0|
|6205|   3.0|
|4144|   3.0|
|3591|   3.0|
|2910|   3.0|
|4649|   3.0|
|6491|   3.0|
|5583|   3.0|
|5358|   3.0|
|5316|   3.0|
|2550|   3.0|
|1471|   3.0|
|1494|   3.0|
|3254|   3.0|
|5537|   3.0|
|4484|   3.0|
|4958|   3.0|
|5305|   3.0|
|1500|   3.0|
```

| | |
|---|---|
| 4854 | 3.0 |
| 4083 | 3.0 |
| 6251 | 3.0 |
| 5928 | 3.0 |
| 1195 | 3.0 |
| 7945 | 3.0 |
| 7393 | 3.0 |
| 5829 | 3.0 |
| 938 | 3.0 |
| 6706 | 3.0 |
| 1996 | 3.0 |
| 3052 | 3.0 |
| 3603 | 3.0 |
| 5013 | 3.0 |
| 4256 | 3.0 |
| 2943 | 3.0 |
| 5375 | 3.0 |
| 5497 | 3.0 |
| 3950 | 3.0 |
| 4683 | 3.0 |
| 5461 | 3.0 |
| 2009 | 3.0 |
| 6243 | 3.0 |
| 5702 | 3.0 |
| 2843 | 3.0 |
| 5693 | 3.0 |
| 5084 | 3.0 |
| 7749 | 3.0 |
| 2035 | 3.0 |
| 5586 | 3.0 |
| 5473 | 3.0 |
| 6250 | 3.0 |
| 4862 | 3.0 |
| 2696 | 3.0 |
| 2377 | 3.0 |
| 4028 | 3.0 |