Extra assignment

Flu detection by analyzing social signals like tweets

Purpose:

It is important to detect the occurrence of flu so that public health authorities can act immediately and reduce the impact. For this data to be available there is always 1-2-week delay between diagnosis of the patient and the data to be available. Traditional approach includes collecting the influenza-like illness activity data from medical practices. In this project I present a framework, that monitors messages posted on Twitter with a mention of flu indicators that tracks and predicts the emergence and spread of a flu in common people.

Apache Spark and IBM Bluemix:

Apache Spark is an open source big data analytics tool. It is used to process large datasets. As Spark provides multi-stage in-memory primitives that can be several times faster for certain applications, it is a viable when compared to Hadoop for certain applications.
Bluemix is a cloud platform as a service (PaaS) developed by IBM. It supports several languages Java, Node.js, PHP, Python, Scala etc.

Procedure:

Using hash tags from user posts on *Twitter* as our input data, we collate and chart the occurrence of keywords. The first instance consists of a problem statement that collects the information of the illness like cold, fever and flu in a given location and time are inferred from the content of *tweets*. The second one is a plot of a graph that uses the tweets collected.
Having experience with collecting and analyzing twitter data in one of the labs, it is advantageous to use the obtained knowledge to create an application in IBM Bluemix.

Steps involved:

1. Configuring the twitter 4J and Watson tone analyzer.
2. Collecting tweets for 30 minutes
3. Storing the results in the parquet file along with the sentiment score
4. Read through the parquet file and generate the graphs

References:

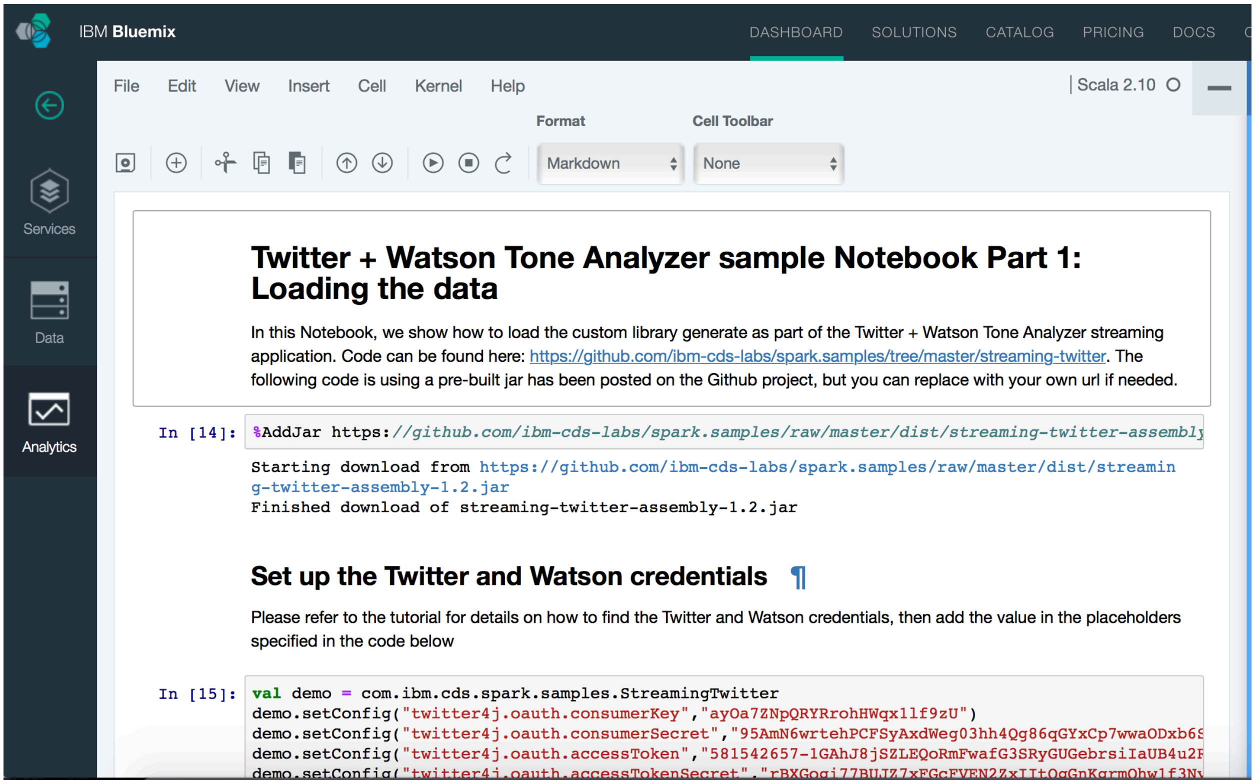
Twitter Tone Analyzer using Apache Spark

Spark Example

OAuth Credentials:

Consumer Secret	95AmN6wrtehPCFSyAxdWeg03hh4Qg86qGYxCp7wwaODxb6Sj82
Owner	SKangokar
Owner ID	581542657
Consumer Key (API Key)	ayOa7ZNpQRYRrohHWqx1lf9zU

Screen Shots:



Twitter + Watson Tone Analyzer sample Notebook Part 1: Loading the data

In this Notebook, we show how to load the custom library generate as part of the Twitter + Watson Tone Analyzer streaming application. Code can be found here: https://github.com/ibm-cds-labs/spark.samples/tree/master/streaming-twitter. The following code is using a pre-built jar has been posted on the Github project, but you can replace with your own url if needed.

In [14]:

```
%AddJar https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-twitter-assembly
```

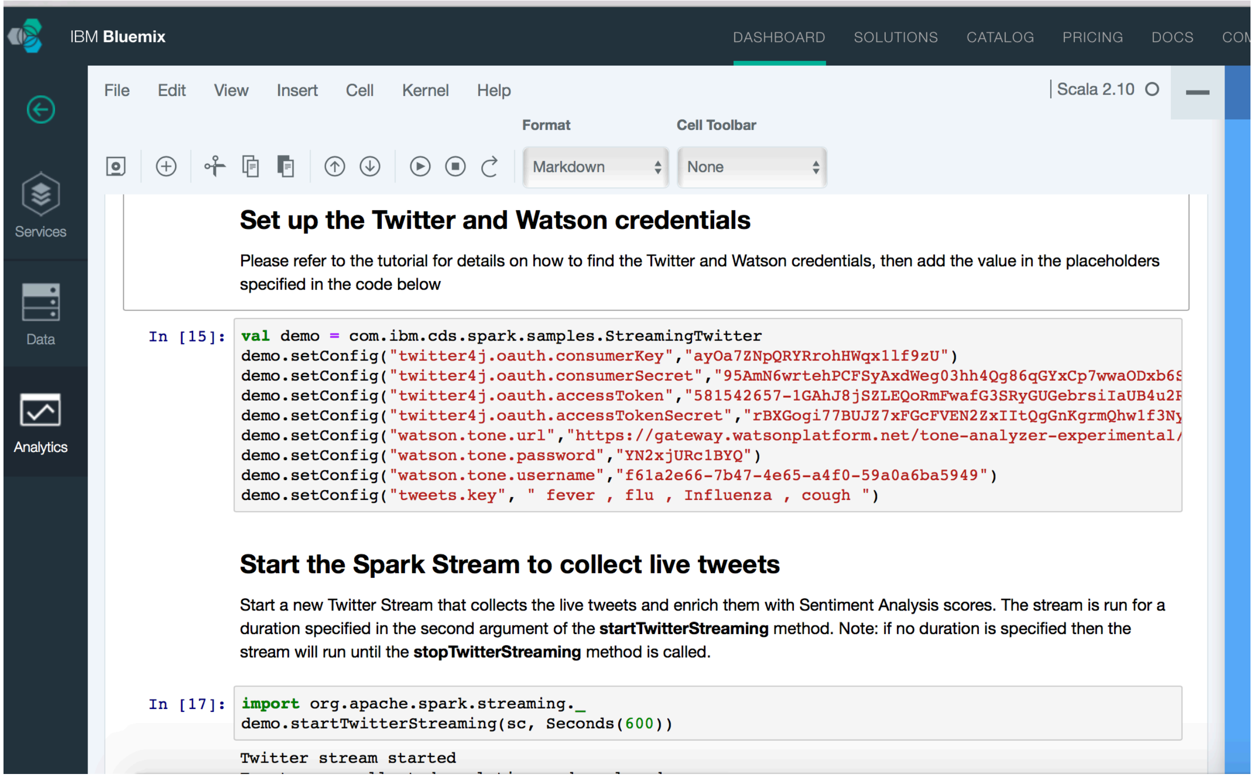
```
Starting download from https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streamin
g-twitter-assembly-1.2.jar
Finished download of streaming-twitter-assembly-1.2.jar
```

Set up the Twitter and Watson credentials ¶

Please refer to the tutorial for details on how to find the Twitter and Watson credentials, then add the value in the placeholders specified in the code below

In [15]:

```
val demo = com.ibm.cds.spark.samples.StreamingTwitter
demo.setConfig("twitter4j.oauth.consumerKey","ayOa7ZNpQRYRrohHWqx1lf9zU")
demo.setConfig("twitter4j.oauth.consumerSecret","95AmN6wrtehPCFSyAxdWeg03hh4Qg86qGYxCp7wwaODxb6S
demo.setConfig("twitter4j.oauth.accessToken","581542657-1GAhJ8jSZLEQoRmFwafG3SRyGUGebrsiIaUB4u2R
demo.setConfig("twitter4j.oauth.accessTokenSecret","rBXGogi77BUJZ7xFGcFVEN2ZxIItOgGnKgrmOhw1f3Ny
```



Set up the Twitter and Watson credentials

Please refer to the tutorial for details on how to find the Twitter and Watson credentials, then add the value in the placeholders specified in the code below

In [15]:

```
val demo = com.ibm.cds.spark.samples.StreamingTwitter
demo.setConfig("twitter4j.oauth.consumerKey","ayOa7ZNpQRYRrohHWqx1lf9zU")
demo.setConfig("twitter4j.oauth.consumerSecret","95AmN6wrtehPCFSyAxdWeg03hh4Qg86qGYxCp7wwaODxb6S
demo.setConfig("twitter4j.oauth.accessToken","581542657-1GAhJ8jSZLEQoRmFwafG3SRyGUGebrsiIaUB4u2F
demo.setConfig("twitter4j.oauth.accessTokenSecret","rBXGogi77BUJZ7xFGcFVEN2ZxIItQgGnKgrmQhw1f3Ny
demo.setConfig("watson.tone.url","https://gateway.watsonplatform.net/tone-analyzer-experimental/
demo.setConfig("watson.tone.password","YN2xjURc1BYQ")
demo.setConfig("watson.tone.username","f61a2e66-7b47-4e65-a4f0-59a0a6ba5949")
demo.setConfig("tweets.key", " fever , flu , Influenza , cough ")
```

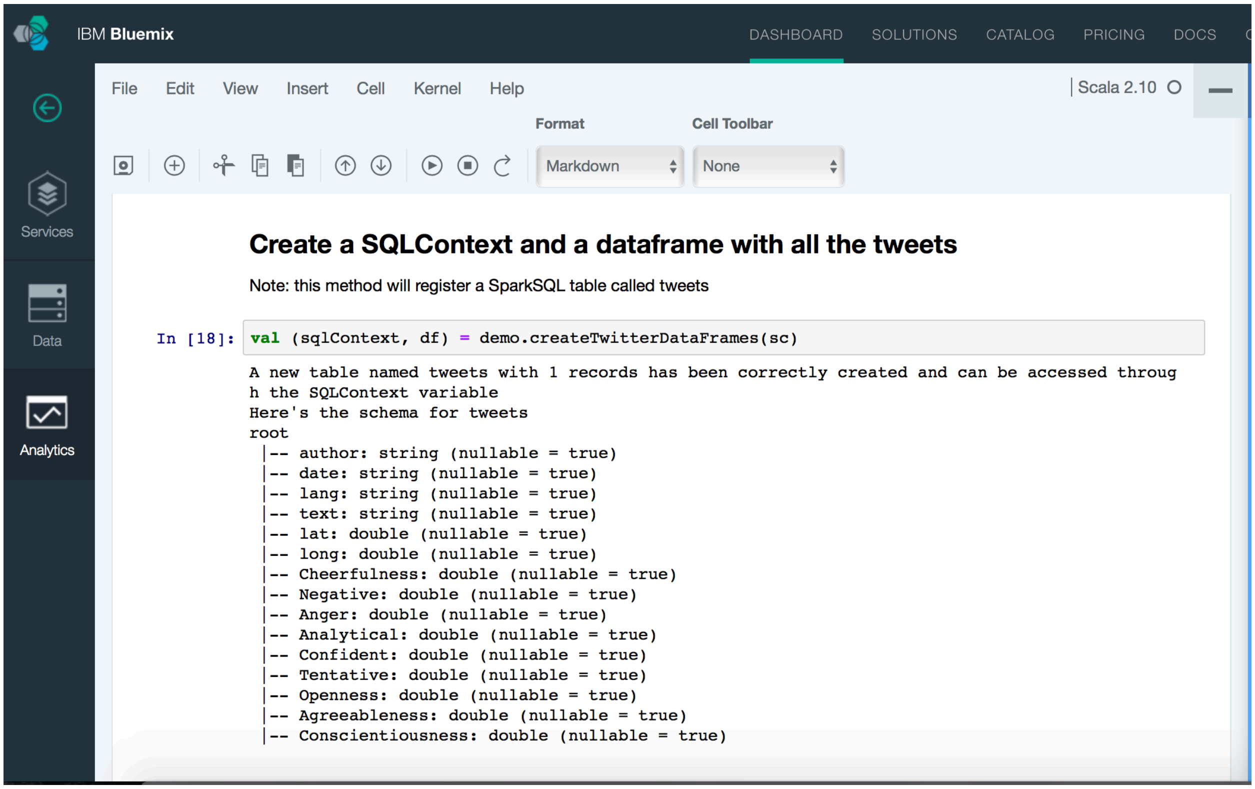
Start the Spark Stream to collect live tweets

Start a new Twitter Stream that collects the live tweets and enrich them with Sentiment Analysis scores. The stream is run for a duration specified in the second argument of the **startTwitterStreaming** method. Note: if no duration is specified then the stream will run until the **stopTwitterStreaming** method is called.

In [17]:

```
import org.apache.spark.streaming._
demo.startTwitterStreaming(sc, Seconds(600))
```

```
Twitter stream started
```



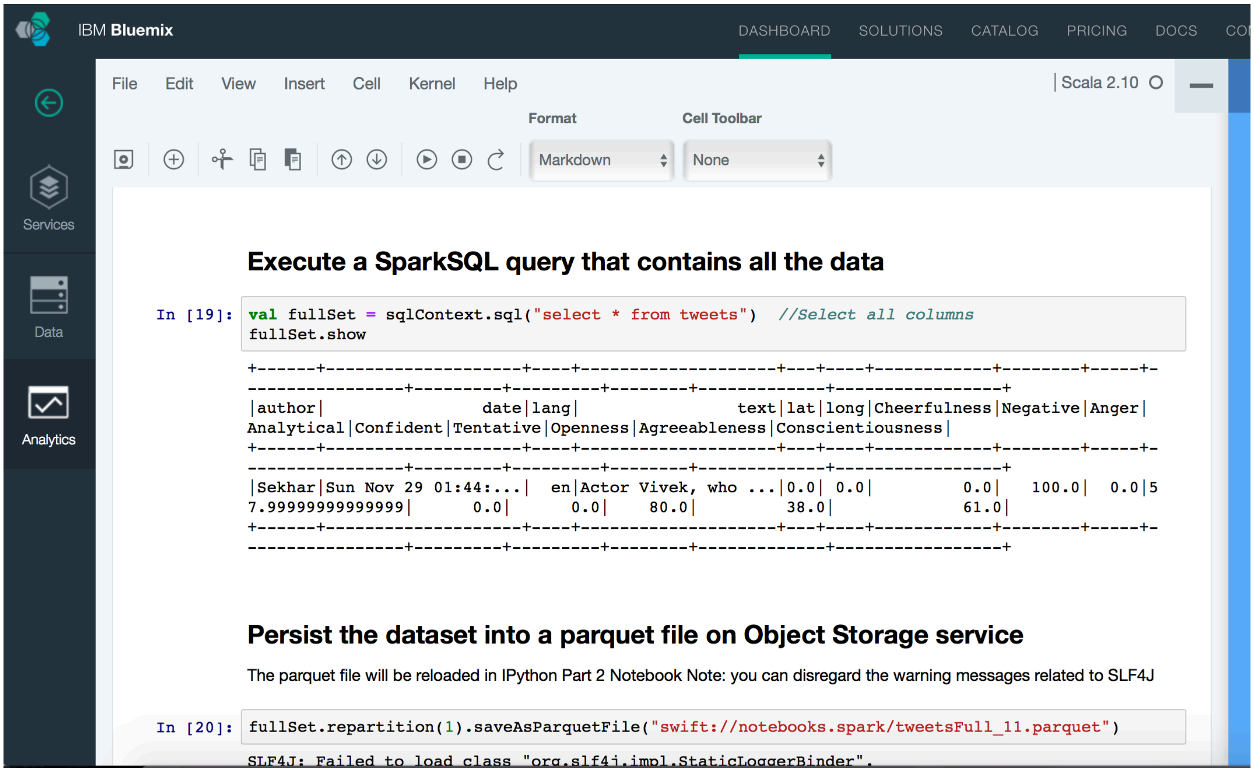
Create a SQLContext and a dataframe with all the tweets

Note: this method will register a SparkSQL table called tweets

In [18]:

```
val (sqlContext, df) = demo.createTwitterDataFrames(sc)
```

```
A new table named tweets with 1 records has been correctly created and can be accessed throug
h the SQLContext variable
Here's the schema for tweets
root
 |-- author: string (nullable = true)
 |-- date: string (nullable = true)
 |-- lang: string (nullable = true)
 |-- text: string (nullable = true)
 |-- lat: double (nullable = true)
 |-- long: double (nullable = true)
 |-- Cheerfulness: double (nullable = true)
 |-- Negative: double (nullable = true)
 |-- Anger: double (nullable = true)
 |-- Analytical: double (nullable = true)
 |-- Confident: double (nullable = true)
 |-- Tentative: double (nullable = true)
 |-- Openness: double (nullable = true)
 |-- Agreeableness: double (nullable = true)
 |-- Conscientiousness: double (nullable = true)
```



Execute a SparkSQL query that contains all the data

```
In [19]: val fullSet = sqlContext.sql("select * from tweets")  //Select all columns
         fullSet.show
```

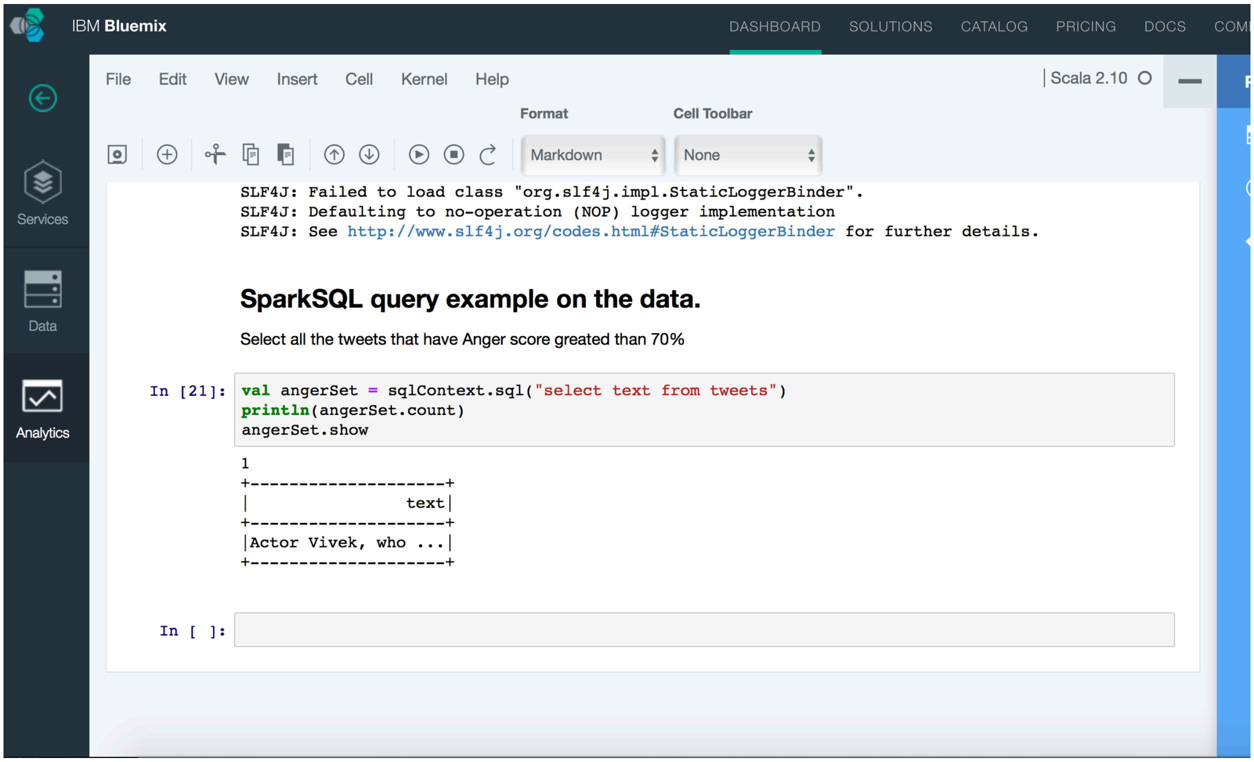
```
+------+--------------------+----+--------------------+---+----+------------+--------+-----+-
----------------+---------+---------+--------+-------------+-----------------+
|author|                date|lang|                text|lat|long|Cheerfulness|Negative|Anger|
Analytical|Confident|Tentative|Openness|Agreeableness|Conscientiousness|
+------+--------------------+----+--------------------+---+----+------------+--------+-----+-
----------------+---------+---------+--------+-------------+-----------------+
|Sekhar|Sun Nov 29 01:44:...|  en|Actor Vivek, who ...|0.0| 0.0|         0.0|   100.0|  0.0|5
7.99999999999999|      0.0|      0.0|    80.0|         38.0|             61.0|
+------+--------------------+----+--------------------+---+----+------------+--------+-----+-
----------------+---------+---------+--------+-------------+-----------------+
```

Persist the dataset into a parquet file on Object Storage service

The parquet file will be reloaded in IPython Part 2 Notebook Note: you can disregard the warning messages related to SLF4J

```
In [20]: fullSet.repartition(1).saveAsParquetFile("swift://notebooks.spark/tweetsFull_11.parquet")
```

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
```



```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

SparkSQL query example on the data.

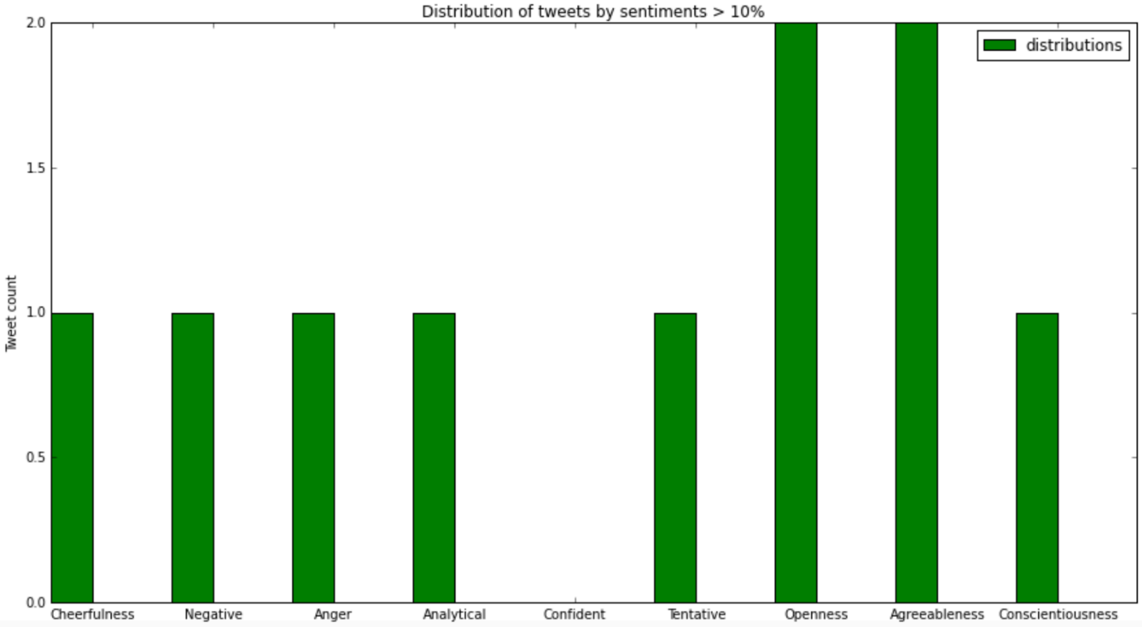
Select all the tweets that have Anger score greated than 70%

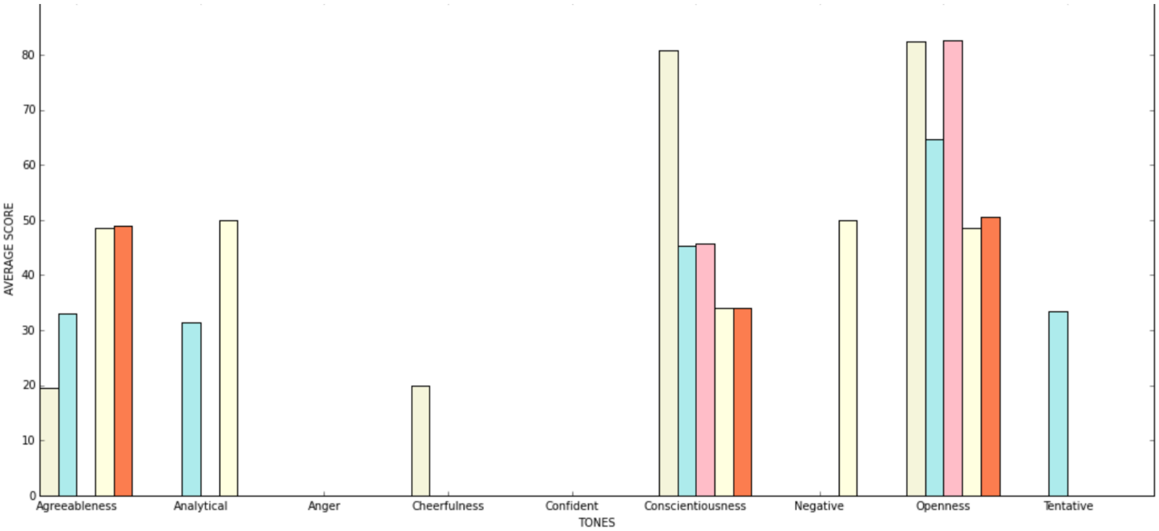
```
In [21]: val angerSet = sqlContext.sql("select text from tweets")
         println(angerSet.count)
         angerSet.show
```

```
1
+--------------------+
|                text|
+--------------------+
|Actor Vivek, who ...|
+--------------------+
```

```
In [ ]:
```

Graph results:




AVERAGE SCORE
0
10
20
30
40
50
60
70
80
Agreeableness
Analytical
Anger
Cheerfulness
Confident
Conscientiousness
Negative
Openness
Tentative
TONES