[Github link](https://github.com/Carloselrecharlie/BD_assessment.git)

# Introduction

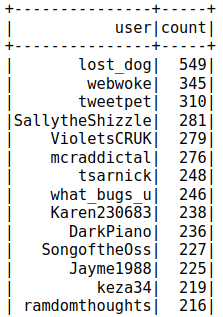
When it comes to Apache Spark there are three logical abstractions to handle data. RDDs (Resilient Distributed Datasets) used to be the only option, they are immutable distributed collections of elements of data and they are useful when a dealing with unstructured data or when, for instance, a developer wants a low-level transformation and actions and control on the dataset. In this case tweets are semi-structured data and the tweets themselves (text) are unstructured data.

DataFrames also are immutable distributed collections of data, but they are organized into named columns, like a table in a relational database and are designed to make large data sets processing easier (data manipulation, filtering, aggregations, and transformations). They allow users to impose a structure onto a distributed collection of data, allowing higher-level abstraction (Damji 2016). Considering these facts the choice was to use DataFrames. Additionally, they also offer better performance optimizations like Project Tungsten and Catalyst optimizer (NK 2023), which can lead to faster data processing compared to RDDs.

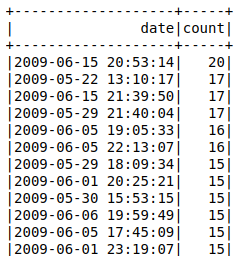
Something significant to point out is that these logical abstractions are not mutually exclusive, actually Dataframes are built on top of the RDDs and can be easily converted into each other. For the purpose of this assignment I also used RDDs at times for didactive reasons, when performance was not degraded. This was because DataFrames are more intuitive and similar to what I already knew. Also used SQL queries for the same reason, even though there was not an actual connection between spark and this DBMS due to the unresolved conflicts with dependencies.

The dataset was taken to Hadoop filesystem and read in to memory as a spark DataFrame, solving the issue with PDT timezone by converting the dates with to\_timestamp and setting legacy timeParserPolicy. Found 1685 duplicate ids (duplicate tweets), confirmed duplicates by aggregating all features except index since it is a unique parameter. Once the duplicates were removed there were 1,598,315 rows remaining.

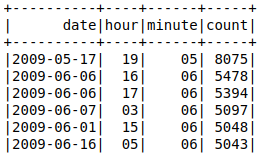
The flag column has one single unique value (NO\_QUERY) which means it does not add anything to the analysis. 659,775 unique users, being the ones with greatest count of tweets as per below:



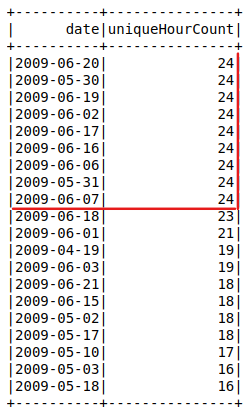
The collection of tweets covers around 2 months and a half in 2009, from 07/04 to 25/06 and the greatest count of tweets per timestamp (HH:MM:SS) is 20. Top counts below:



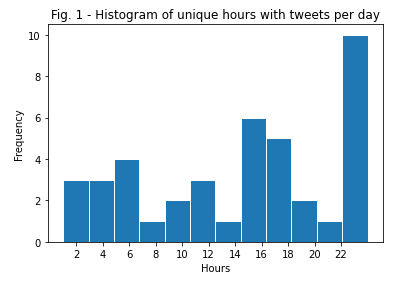
Regarding natural minutes, the greatest count of tweets is above 8 thousand:



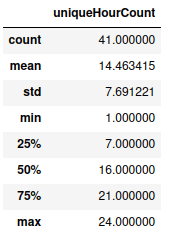
In order to plan the time series, the analysis of the timestamps showed that only 10 days have tweets in each of the 24 hours which make up a day. Considering this, an hourly time series would imply an excessive amount of imputation.



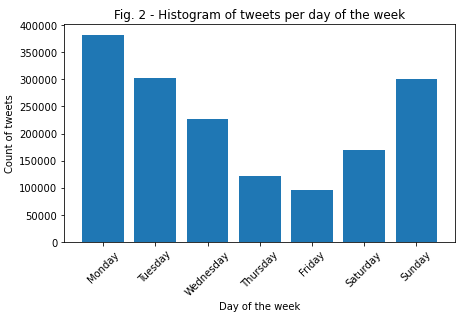
Same as a histogram:



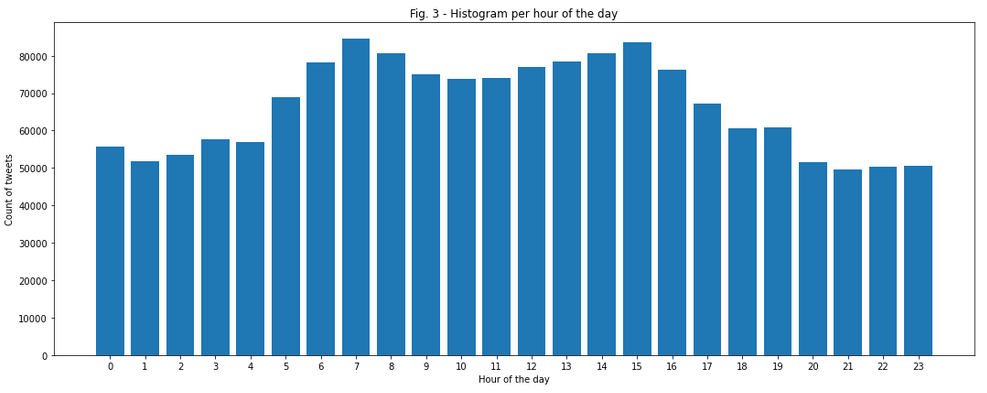
The percentiles of the counts above showed that within the 2 and a half months (with timestamps from 80 days) there are just 41 dates present, being the rest missing. Again this represents a significant amount of imputation. Also noticeable how half of the dates present hold tweets from up to 16 hours.



Sunday, Monday and Tuesday are the days of the week with greater count of tweets:

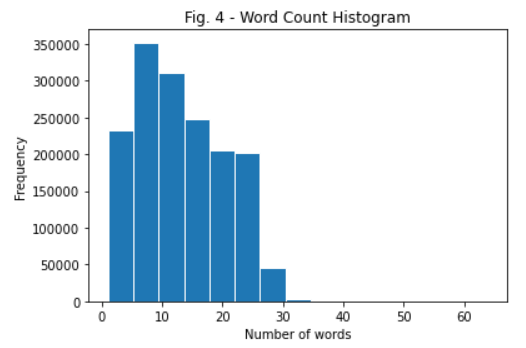


Approximately business hours (UTC due to the time conversion applied) the count of tweets is greater:



## Exploration of text feature

There are 21,052,270 words in the data set with an average of 13.2 words per tweet and a standard deviation of 6.9. The average of characters per tweet is 74. The histogram below showed most tweet have around 10 words:



# References

Damji, J. (2016) *RDD vs DataFrames and datasets: A tale of three apache spark apis*, *Databricks*. Available at: https://www.databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html (Accessed: 17 July 2023).

*Datetime patterns for formatting and parsing* (no date) *Datetime patterns - Spark 3.4.1 Documentation*. Available at: https://spark.apache.org/docs/latest/sql-ref-datetime-pattern.html (Accessed: 13 July 2023).

NK, N. (2023) *Spark Performance Tuning & Best Practices*, *Spark By {Examples}*. Available at: https://sparkbyexamples.com/spark/spark-performance-tuning/?expand\_article=1 (Accessed: 2 July 2023).