[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 f.i. a developer want 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 this the choice would be 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.

However 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 for didactive reasons, since dataFrames are more intuitive and similar to what I already knew.

# 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).

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).