CCT College Dublin

|  |  |
| --- | --- |
| Module Title: | Advanced Data Analysis |
| Assessment Title: | Decoding Twitter Emotions: Uncovering Sentiment Patterns and Extracting Insights |
| Lecturer Name: | David McQuaid |
| Student Full Name: | Haider Riaz Ghuman |
| Student Number: | 2022564 |
| Assessment Due Date: | 07/01/2024 |
| Date of Submission: | 07/01/2024 |

Declaration

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

https://github.com/342406/ProjectTweets

Table of Contents

[Introduction 4](#_Toc155549068)

[Problem Statement 4](#_Toc155549069)

[Objective 4](#_Toc155549070)

[Dataset Overview 5](#_Toc155549071)

[Steps Involved in Comprehensive Analysis 5](#_Toc155549072)

[Step 1: Data Ingestion into NoSQL Storage 5](#_Toc155549073)

[- Initializing the path of the essential big data module and creating a spark session. 6](#_Toc155549074)

[- Reading and Saving the Data into Mongo Db: 6](#_Toc155549075)

[- Database View: 7](#_Toc155549076)

[Step 2: Big Data Processing for Dataset Preprocessing 7](#_Toc155549077)

[- Remove Duplicates and Handle Null Values: 8](#_Toc155549078)

[- Dealing with Date-Time Data: 8](#_Toc155549079)

[- Text Cleaning for sentiments analysis: 9](#_Toc155549080)

[- View of cleaned and uncleaned tweets 9](#_Toc155549081)

[- Feature Selection: 9](#_Toc155549082)

[Step 3: Dual Storage Strategy and Performance Evaluation 10](#_Toc155549083)

[Workload A, Execution and Interpretation: 10](#_Toc155549084)

[- Overall Run Read/Update: 11](#_Toc155549085)

[- Throughput (ops/sec) analysis: 12](#_Toc155549086)

[- Throughput analysis Load/Insert: 12](#_Toc155549087)

[- Average Latency during Read/Update for MongoDB and MySQL: 13](#_Toc155549088)

[- Result: 13](#_Toc155549089)

[Step 4: Sentiment Analysis Implementation: 14](#_Toc155549090)

[- Using the built-in methods for calculating the *subjectivity* and *polarity*. 14](#_Toc155549091)

[- Setting the Labels based on values. 14](#_Toc155549092)

[- Final view of the dataset with sentiment labels 15](#_Toc155549093)

[- Plotting the sentiment of people over time on Twitter, using the Plotly library. 15](#_Toc155549094)

[- Result in Sentiment Overtime 16](#_Toc155549095)

[- Plotting the sentiment of people over the week on Twitter 16](#_Toc155549096)

[- Plotting the sentiment of people over Quarters on Twitter 17](#_Toc155549097)

[Step 5: Creation of an Interactive Dashboard 18](#_Toc155549098)

[Conclusion: 19](#_Toc155549099)

[Key Observations: 19](#_Toc155549100)

[References 20](#_Toc155549101)

Decoding Twitter Emotions:

Uncovering Sentiment Patterns and Extracting Insights.

# Introduction

Welcome to this comprehensive report detailing our analysis of a complex social media dataset containing 16 million tweets. In this rich dataset, we are harnessing advanced analytics for gleaning meaningful insights by utilizing the power of big data technologies such as Hadoop MapReduce or Spark and also comparative analysis of MySQL and MongoDB performance, supported by quantitative analysis.

This report unravels the intricacies of data storage and processing, explains our choices, and offers a comprehensive analysis of databases, sentiment changes, and future forecasts, encapsulating the essence of our big data analytics exploration in the realm of social media.

# Problem Statement

Understanding public sentiment is crucial for various applications, including market research, reputation management, and policy-making. However, manually analyzing a large volume of tweets is time-consuming and impractical. Therefore, automated sentiment analysis techniques, such as machine learning and natural language processing, are needed to efficiently process and analyze Twitter data, providing valuable insights into public sentiment.

Social media users generated a massive amount of data over the past few years, with a significant portion coming from social media platforms (Twitter, YouTube, Facebook, etc.) These platforms serve as a hub for people to share pictures, status updates, and thoughts, making the data obtained from them highly valuable for market research and understanding people's thoughts.

In our study, we have taken 1 million time-series tweets for sentiment analysis and uncover hidden patterns. By leveraging machine learning and natural language processing, we aim to extract meaningful insights from this vast amount of data.

# Objective

To explore the concept of sentiment analysis on Twitter and its relevance in understanding public emotions and opinions. This involves reviewing previous studies and research that have focused on sentiment analysis techniques specifically applied to Twitter time series data.

Applying natural language processing (NLP) and machine learning techniques for sentiment analysis on a large dataset of Twitter data also includes preprocessing data using big data techniques and storing data in a NoSQL database or SQL database. The objective is to classify tweets into different sentiment categories, such as positive, negative or neutral, and evaluate the effectiveness of the chosen techniques in capturing Twitter emotions.

To uncover sentiment patterns and trends within the analyzed Twitter data. By analyzing the classified tweets, the aim is to identify common themes, popular sentiments, and any significant shifts in sentiment over time. This step involves visualizations, statistical analysis, and other exploratory techniques.

# Dataset Overview

My analysis is centred around a robust and intricate dataset that provides a nuanced perspective on social media users.

This dataset encapsulates a wealth of information, including attributes such as:

[“ID”, “Date”, “Flag”, “User”, “Text”],

Contains 1.600.000 tweets, extracted using the Twitter API.

These attributes construct a comprehensive portrayal of social media users, facilitating a deep dive into the evolution of their sentiments over a specific period.

# Steps Involved in Comprehensive Analysis

## Step 1: Data Ingestion into NoSQL Storage

In this step, our main target is to create a spark session and read them from the given “ProjectTweets.csv” file containing the 16000000 Tweets. Now, we have selected Mongo DB as a NoSQL database and Spark as big data technology.

**The reason,** to use Mongo as our database for storing Twitter data is because of its proficiency in handling large-scale data storage. MongoDB's capability to manage vast volumes of data makes it well-suited for our needs. Its collection-wise storage and document-oriented approach are particularly beneficial for handling textual data efficiently

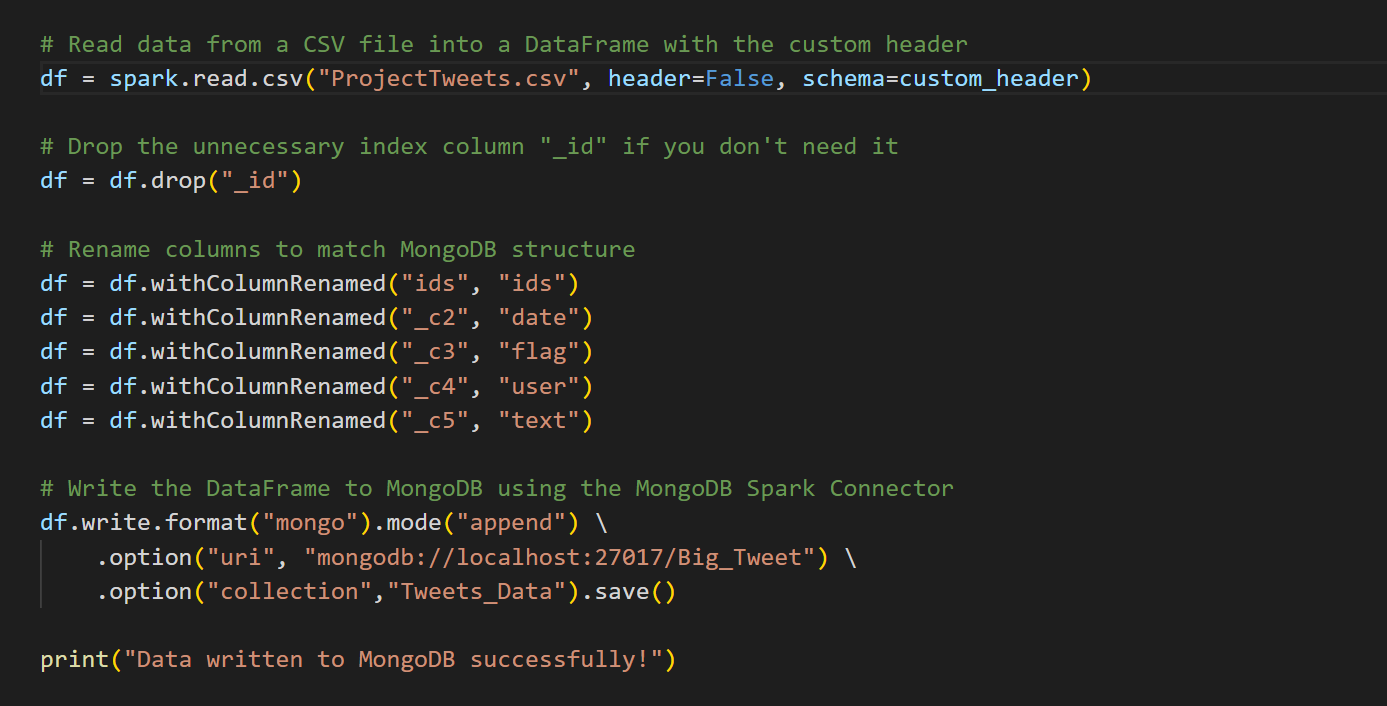
**The reason,** to prefer *Python* sentiment analysis on Twitter data is due to its extensive libraries (e.g., TextBlob, NLTK), rich ecosystem for data processing, and simplicity in implementing natural language processing tasks.

**The reason,** to use Spark is because its distributed computing capabilities make it an ideal choice for efficiently handling our large Twitter dataset. With parallel processing across a cluster of machines, Spark ensures fast and scalable data analysis. Its fault tolerance features and integration with Python through PySpark enhance reliability and versatility, making it well-suited for the complexities of working with extensive Twitter data.

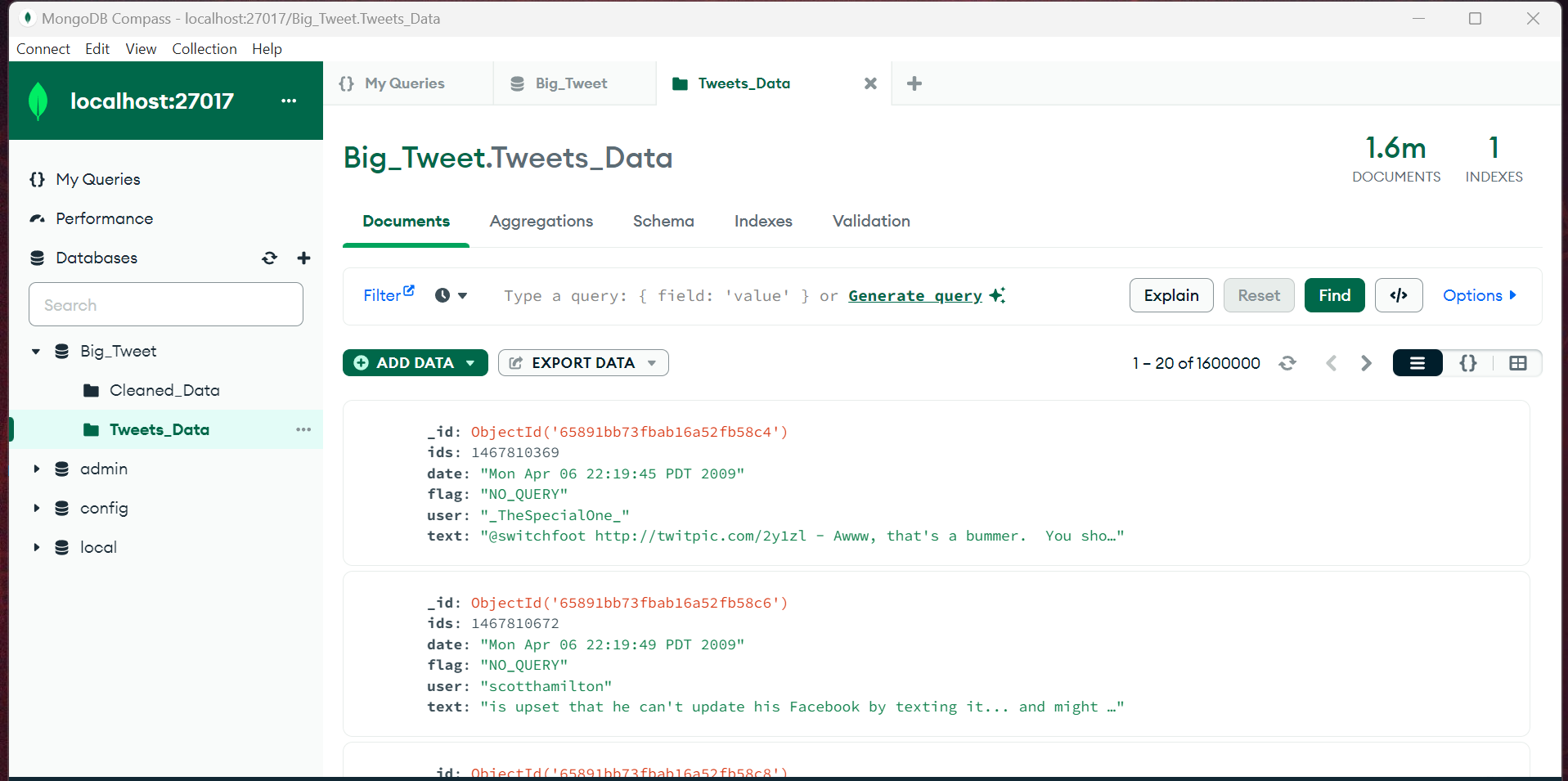
### Initializing the path of the essential big data module and creating a spark session.



### Reading and Saving the Data into Mongo Db:

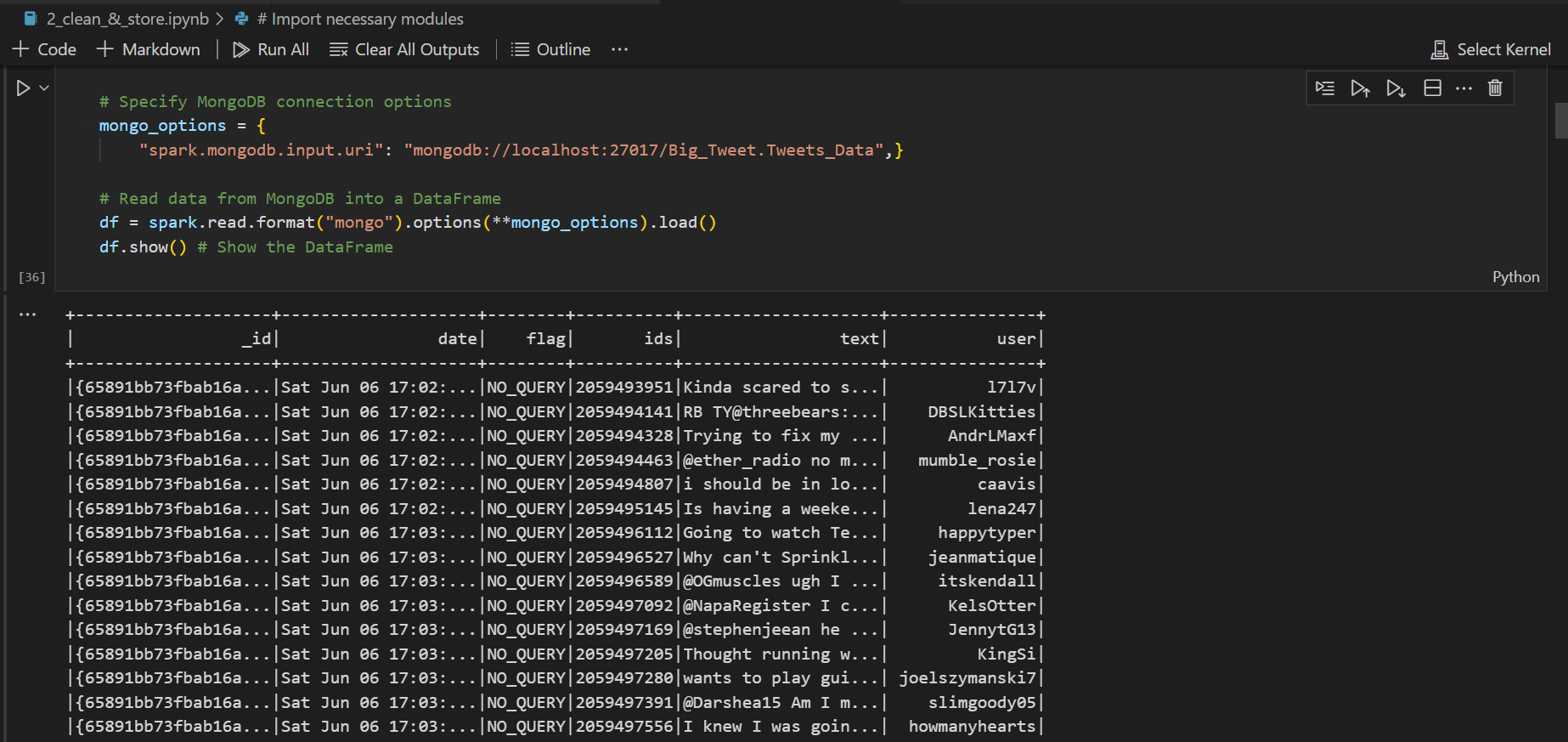


### Database View:



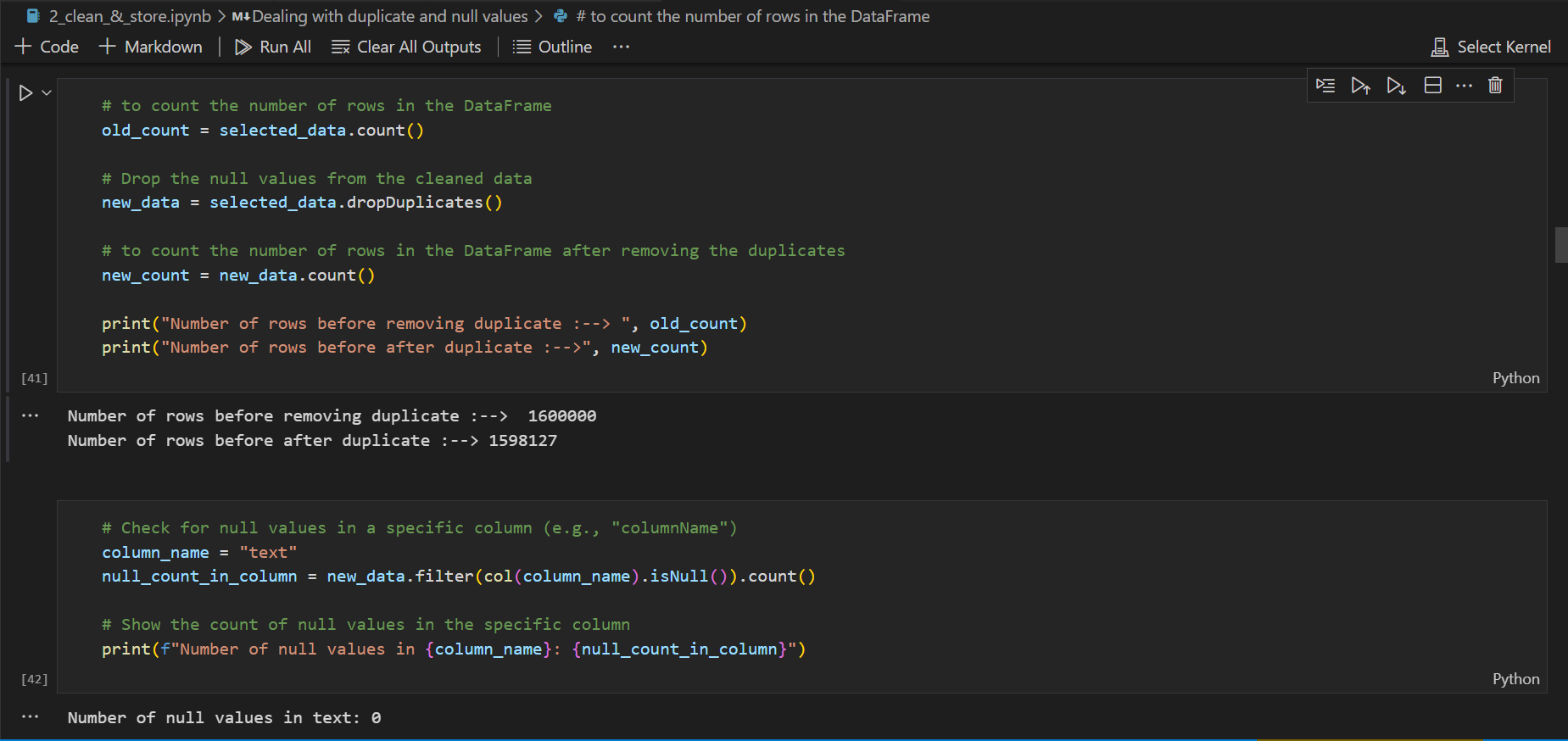
## Step 2: Big Data Processing for Dataset Preprocessing

In this step, we have performed many steps required to improve the overall quality of our data. First, we have read the data from Mongo ***DB*** using ***PySpark.***

******

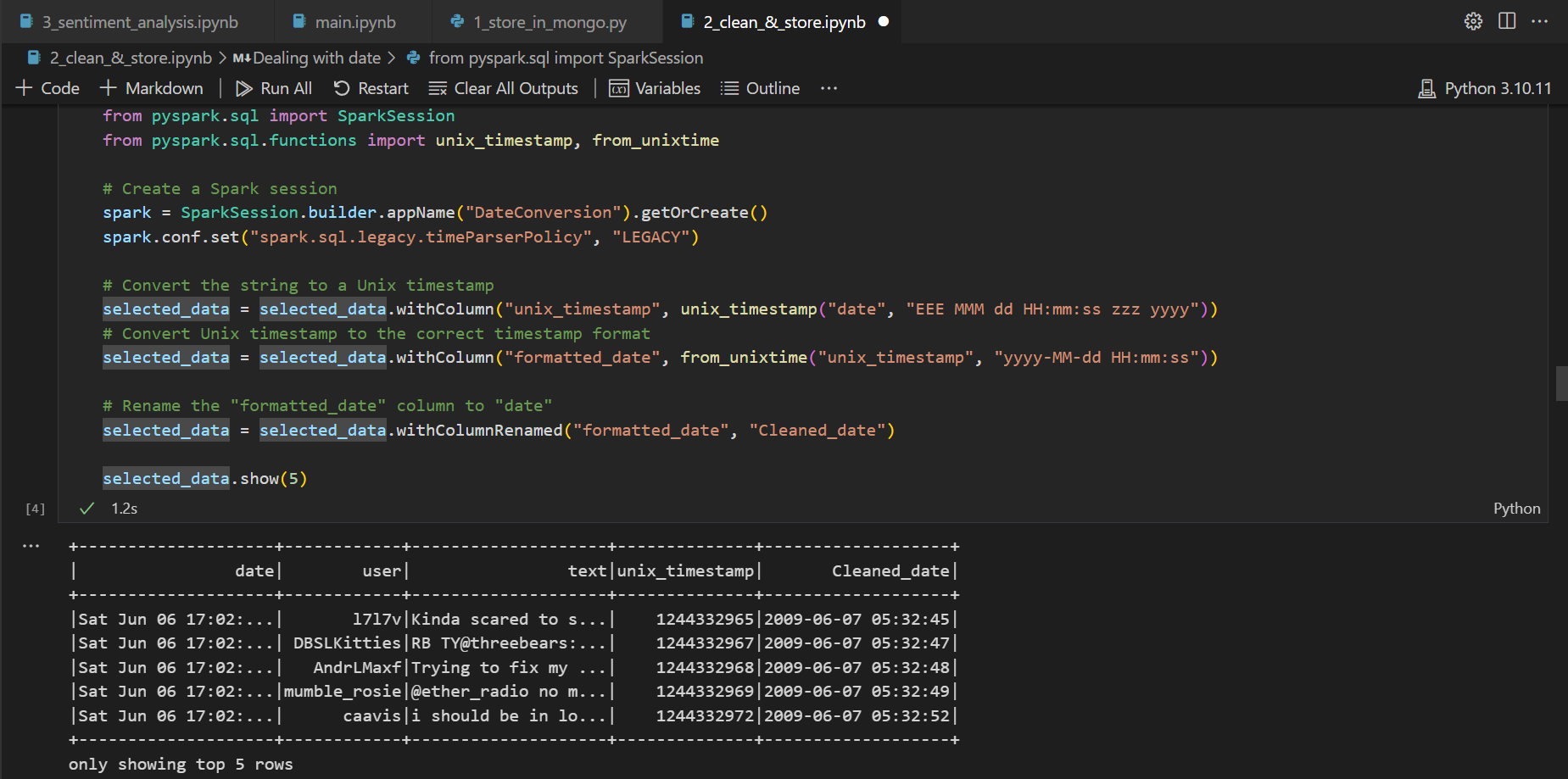
### Remove Duplicates and Handle Null Values:

In the direction of data cleaning to improve the quality of data we have removed repeated entries and dealt with missing information to ensure a more accurate analysis.



### Dealing with Date-Time Data:

As we can see our data columns contain data in long format and are also not treated as date, as it is needed for our time-based changes in social media users so I have also cleaned the date data.

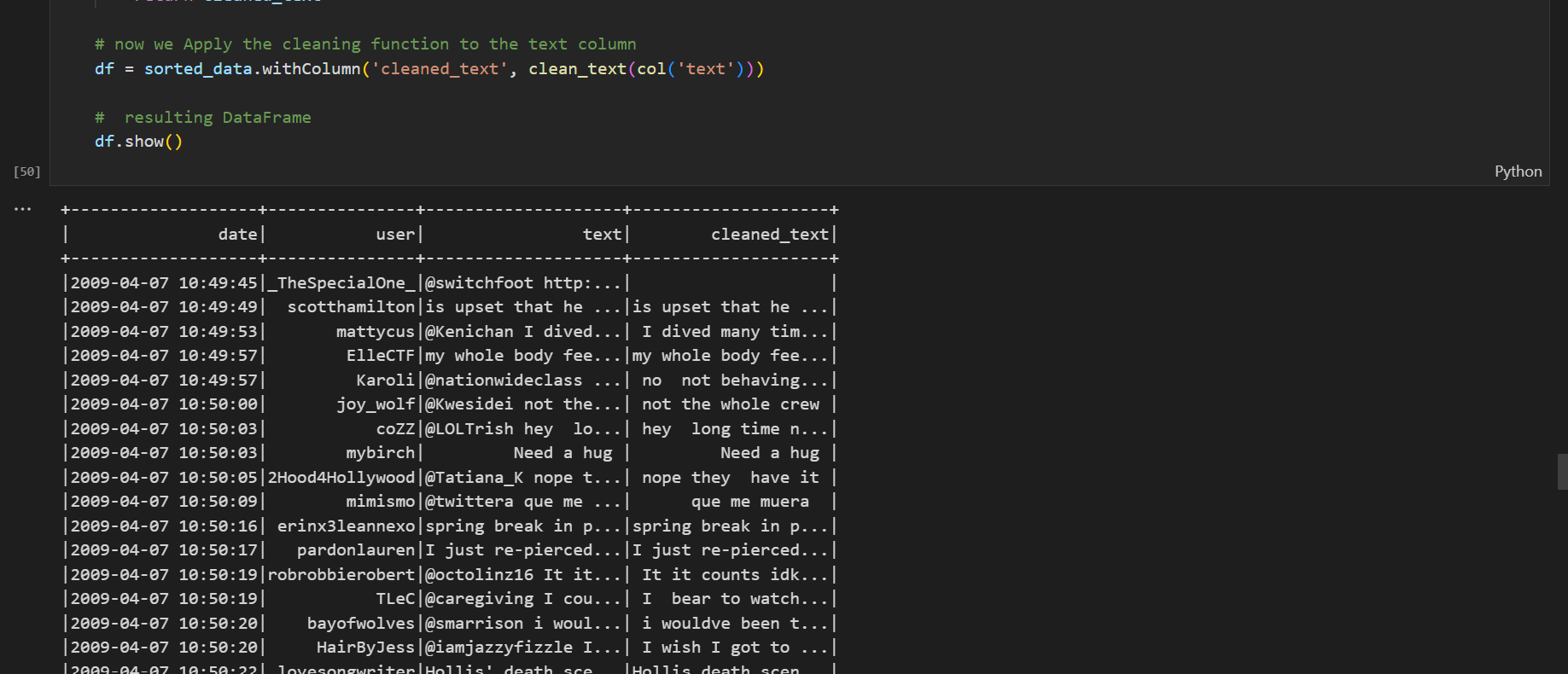


### Text Cleaning for sentiment analysis:

It is an essential step for our data as our data set contains links, tags, grammatical mistakes, special characters, stop words and many more so for our sentiments analysis we need cleaned data, So I refined the dataset by using ***regular expressions*** and leveraging advanced libraries such as ***NLTK, Text Blob, and spacy***. This thorough text-cleaning process not only ensures the quality and consistency of the data but also sets the stage for more accurate and meaningful analysis.

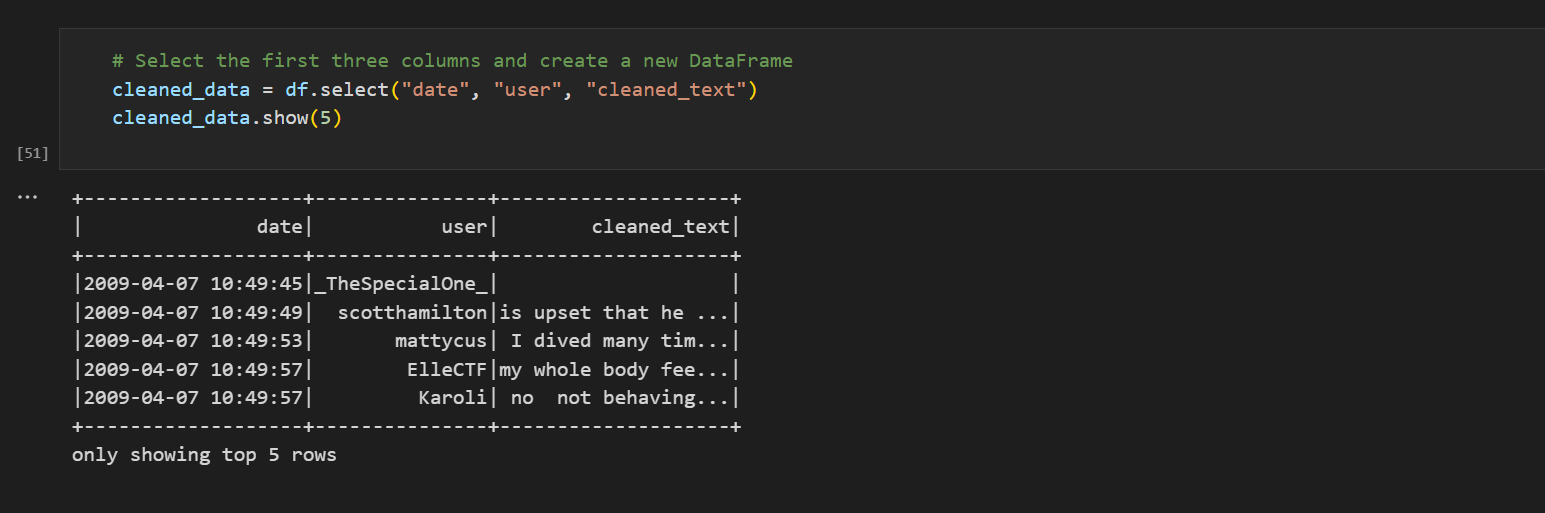
**The reason,** why we are using ***NLTK, Text Blob, and Spacy*** is that these are open-source libraries in Python for processing textual data. It is a powerful package that reduces the complexity of the contextual data and derives in-depth information from the text.

### View of cleaned and uncleaned tweets



### Feature Selection:

Also performed feature selection to choose the most relevant features for the sentiments analysis and streamline the dataset for efficient processing.



## Step 3: Dual Storage Strategy and Performance Evaluation

As we have a large amount of unstructured data and want to perform many complex operations on it it is an essential part to choose a database wisely.

So, to identify the most suitable database we are going to conduct a benchmark comparison using YCSB (which is an open-source benchmarking tool).

The given results are in the form of a chart taken from the referenced article as we had many challenges so that is why we also looked at some other pre-works.

First, we have cloned the YCSB repository into our local system.

Also, make sure that both Mono DB and MySQL are installed in your system.

Then we modified the Mongo Db URL and saved the collection details in to **workload file within** in **workloads** folder.

**The reason, for choosing** workload is mentioned below.

*50% read operations, 50% update operations.*

*Suitable for read-intensive applications where updates are less frequent.*

*Ideal for assessing read performance.*

Then we installed and configured Maven and built a binding for Mongo DB by providing the connectivity details in the config file doing this step was also a challenge.

Also, specify the workload and number of treads.

Then after that, we executed some commands to load the workload and then performed the test.

Note -> The same steps are performed for the MySQL.

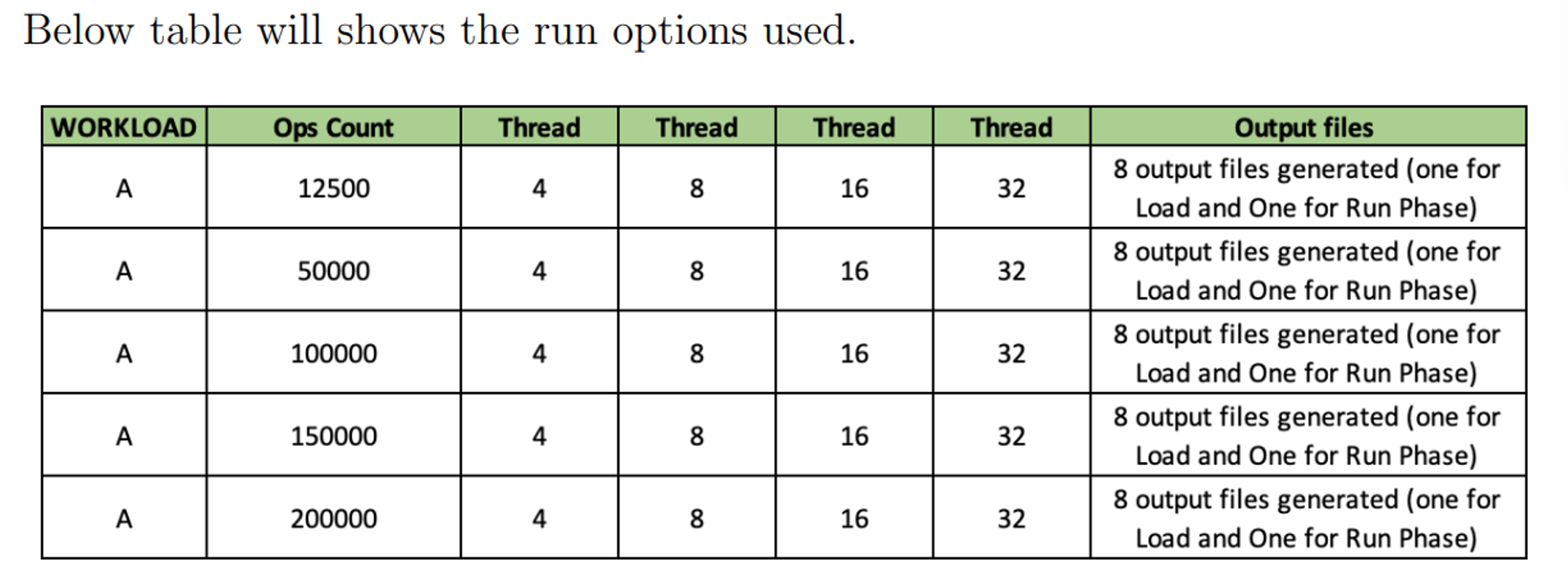
## Workload A, Execution and Interpretation:

Workload A which is an update-heavy workload with an equal mix of (50/50) reads and writes was executed for both databases, MongoDB and MySQL. The details are below and the interpretations of the results follow in the form of graphs plotted for various load and run output parameters. These parameters are vital inputs and allow the user/organization to make informed decisions on the choice of the DB they want to use for their business applications.

• Workload A executed for MySQL with the operation and record count of 12500, 50000, 100000, 150000, 200000

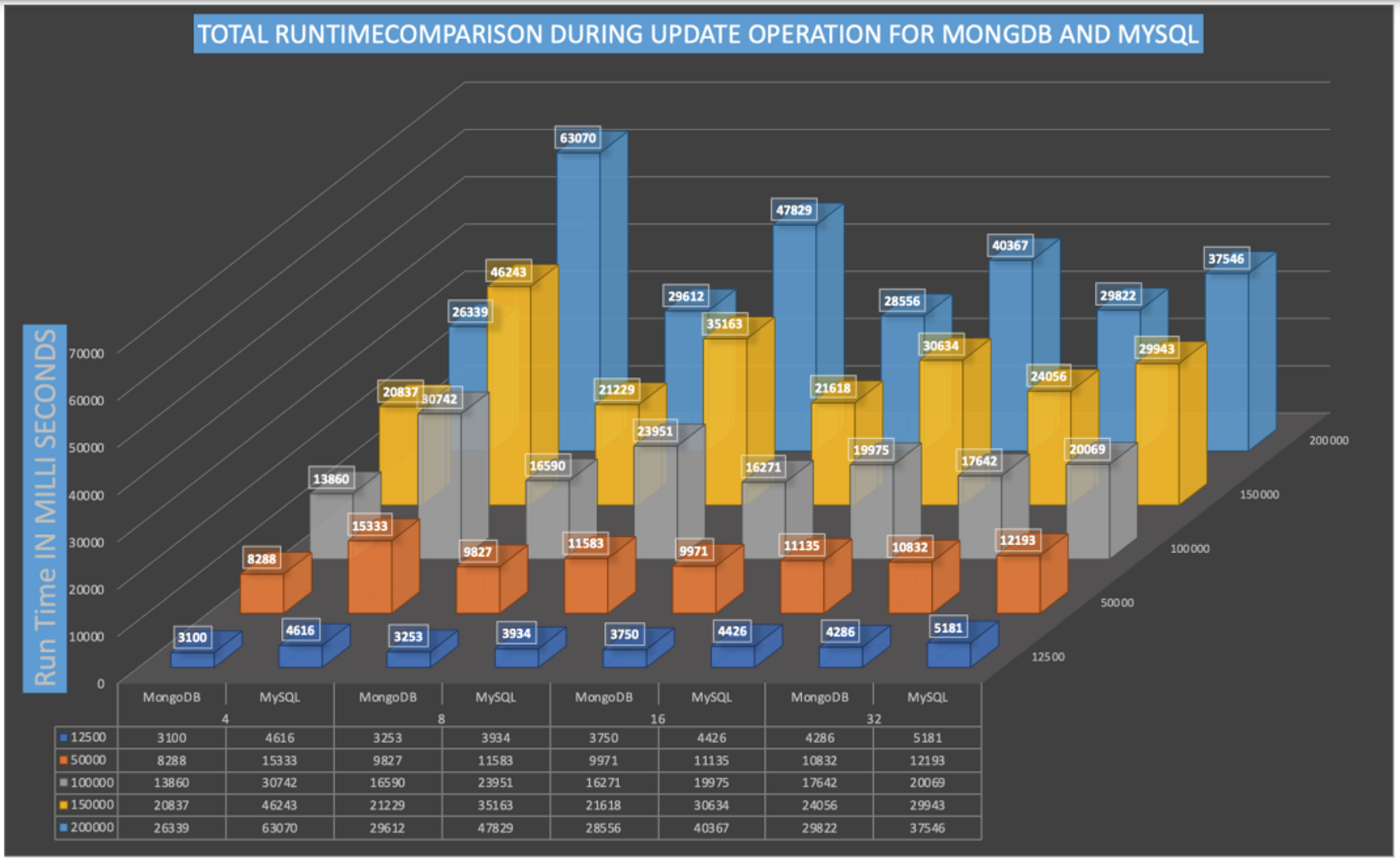
• Workload A executed for MongoDB with the operation and record count of 12500, 50000, 100000, 150000, 200000

• The above workloads were executed with the thread options of 4,8,16,32.



### Overall Run Read/Update:

The graph shows the overall run time of workload A during the Read/Update (Run) operation for both MongoDB and MySQL for the mentioned operations count and threads. As we know Workload A is 50/50 read and write during the load phase the YCSB tool reads and updates the records as mentioned in the ops count and record count parameter and the overall run time is recorded and plotted below.

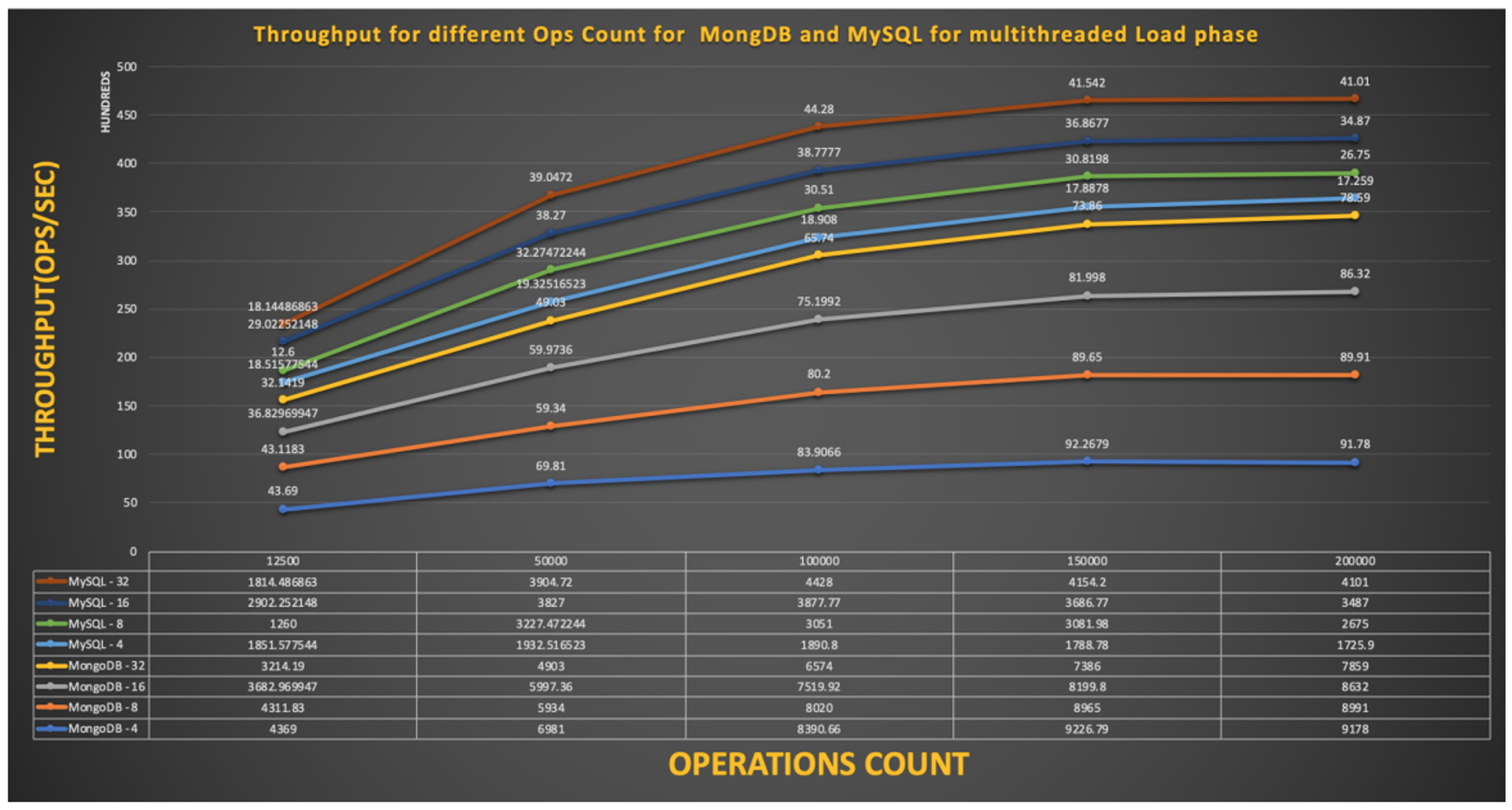


### Throughput (ops/sec) analysis:

In this section, we will analyze the throughput variation in both MySQL and MongoDB for load and run phases with multiple operation counts and threads.

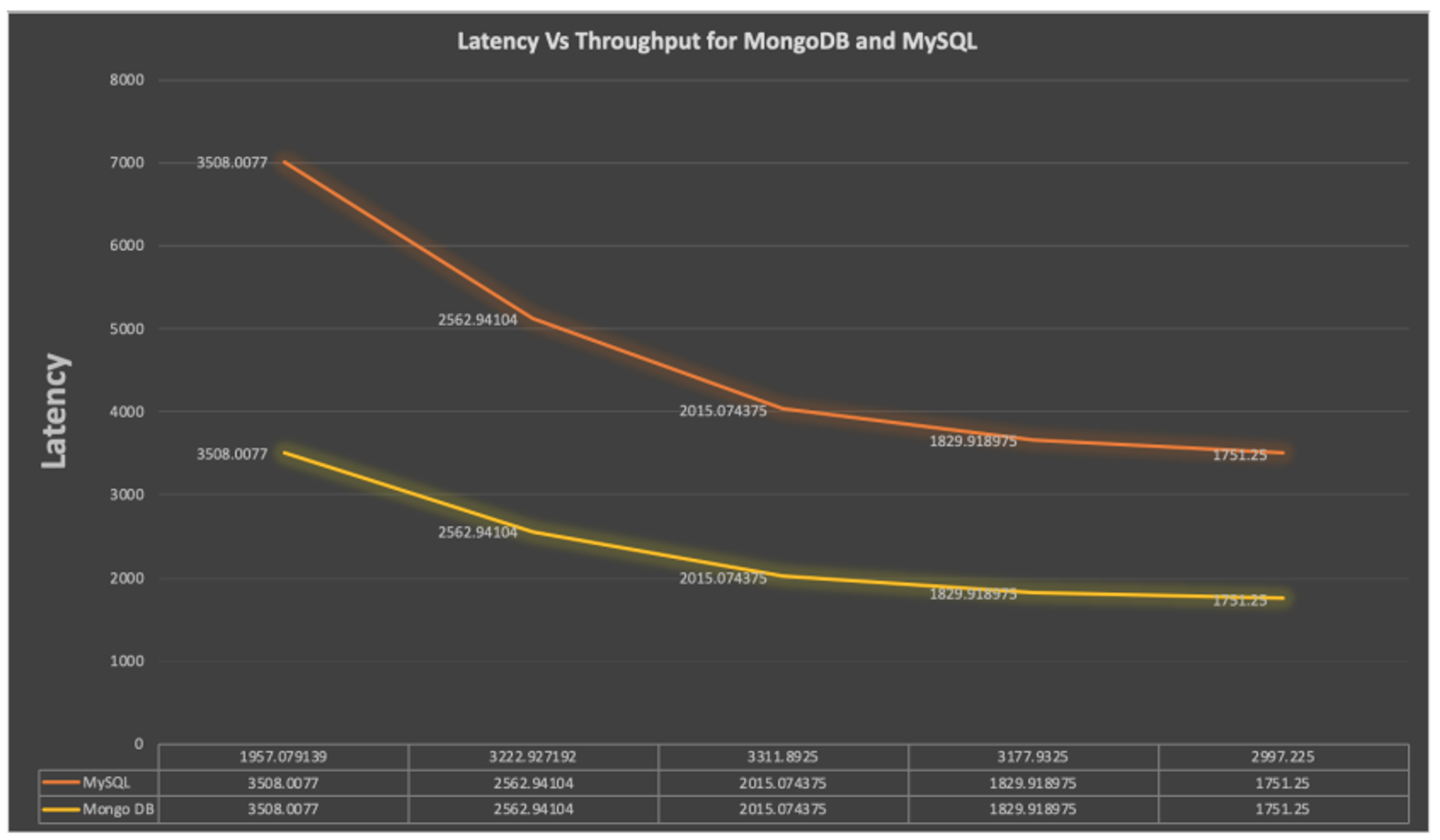
### Throughput analysis Load/Insert:

The graph shows the overall throughput of workload A during the Insert (Load) operation for both MongoDB and MySQL for the mentioned operations count and threads. The throughput for each op count load with options of 4,8,16,32 threads has been executed and the results are recorded and shown in the graph below.



### Average Latency during Read/Update for MongoDB and MySQL:

The graph below shows the average latency for read and update operations for multiple values of ops count for MongoDB and MySQL databases. As we can see from graph 12 the read latency difference between MongoDB and MySQL is not very significant and both perform almost similarly for multiple read requests with varying numbers of threads. Having multiple threaded operations does not impact significantly the read latency and the variation between MongoDB and MySQL is somewhat constant. For 4 threads the difference between the read latency for MySQL and MongoDB is 20% and then for threads 4, 8, and 16, it's 12% approximately. Here also MongoDB is a better performer.

****

### Result:

Our main objective of this analysis was to compare SQL and NOSQL databases to get an output that which database is a suitable choice.

The advantages of MongoDB can be seen from the tests conducted using the YCSB framework, where for each type of workload (A, C, F) it is shown that for all the parameters MongoDB has performed better than MySQL. Especially in terms of Latency and Throughput MongoDB stands out, especially for a higher number of operations.

MySQL is relatively slow because it organizes information logically in tables. The database must write and read data from many tables to update or retrieve information, increasing server load and degrading speed.

MongoDB is the right choice if you are deciding based on higher speed and performance.

## Step 4: Sentiment Analysis Implementation:

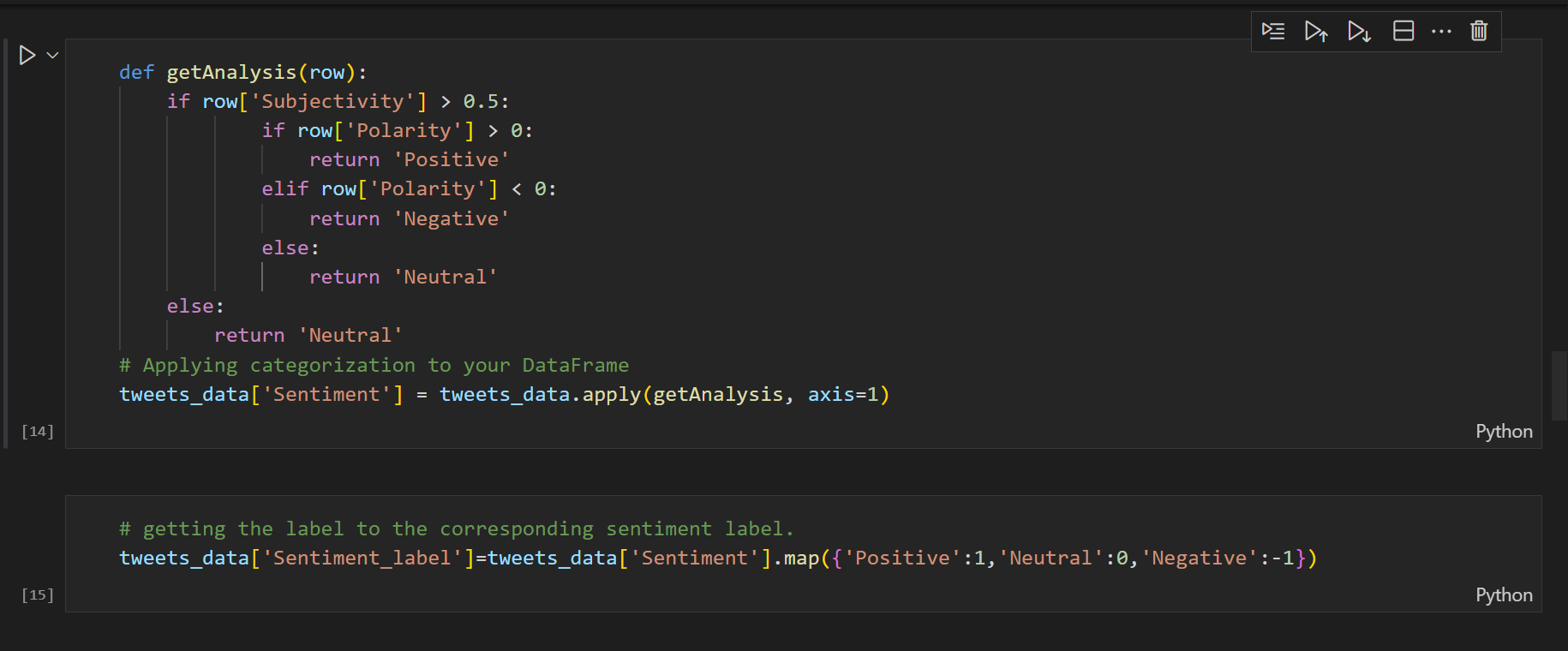
As the final step, we are moving towards the sentiment analysis, which is an essential step of our analysis. So, for that, we are using **TextBlob** (a popular NLP library that simplifies text processing tasks, including sentiment analysis).

**The reason,** why we are using the pre-trained model for sentiment analysis is that our dataset doesn’t contain any target label so we are unable to train our model.

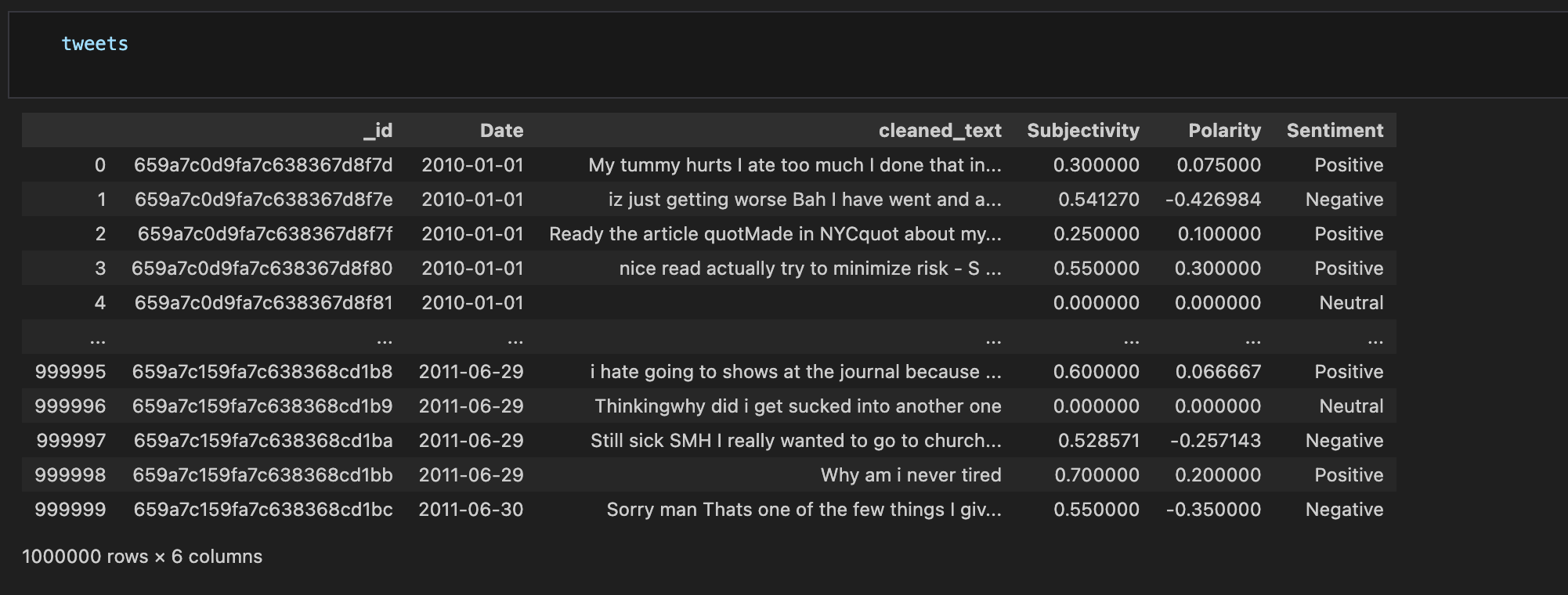
### Using the built-in methods for calculating the *subjectivity* and *polarity*.

****

### Setting the Labels based on values.

****

### Final view of the dataset with sentiment labels



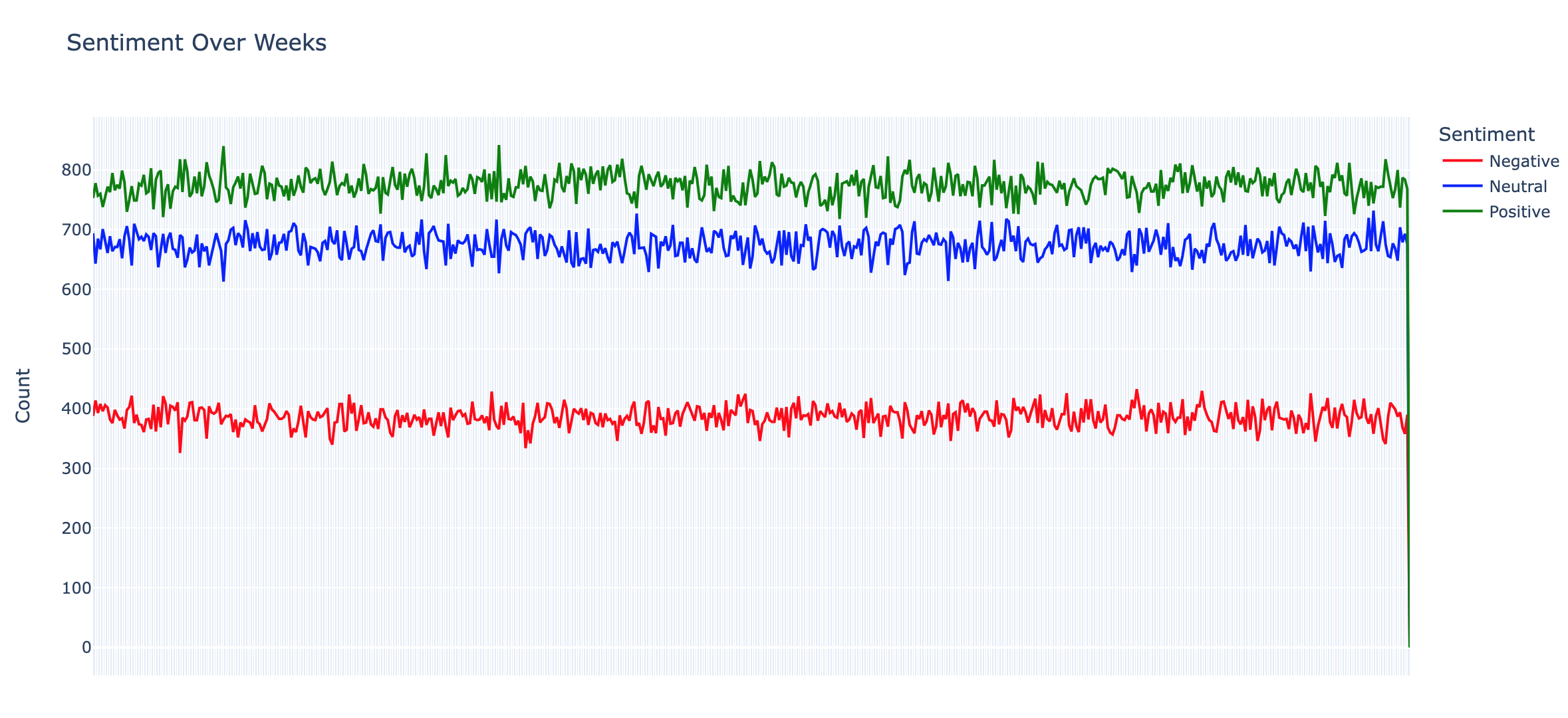
### Plotting the sentiment of people over time on Twitter, using plotly library.



### Result in Sentiment Overtime



### Plotting the sentiment of people over the week on Twitter



### Plotting the sentiment of people over Quarters on Twitter



## Step 5: Creation of an Interactive Dashboard

****

# Conclusion:

The analysis of 1,000,000 tweets using PySpark in the realm of big data technology has provided valuable insights into the sentiment trends within Twitter data. Through the process of data cleaning and sentiment analysis, we categorized the tweets into three main sentiments: positive, negative, and neutral.

Upon visualizing the data through charts, a noteworthy trend emerged, indicating a steady increase in the volume of tweets over time. This growth in Twitter activity may signify a growing user base or increased engagement on the platform.

## Key Observations:

Steady Increase in Tweet Volume:

The data reveals a consistent growth in the number of tweets over time.

Dominance of Neutral Sentiments:

A significant majority (64%) of tweets were classified as neutral, indicating a prevalence of non-opinionated or informational content.

Positive Sentiments:

Positive tweets constituted 24% of the dataset, suggesting a generally favourable sentiment within the Twitter community.

Limited Negative Sentiments:

While present, negative sentiments accounted for 13% of the dataset, indicating a relatively lower prevalence of dissatisfaction or concerns among users.

# References

*Performance benchmarking and comparison of cloud-based databases ...* Available at: https://www.researchgate.net/publication/344047197\_Performance\_Benchmarking\_and\_Comparison\_of\_Cloud\_Based\_Databases\_MongoDB\_NoSQL\_Vs\_MySQL\_Relational\_using\_YCSB (Accessed: 07 January 2024).

*Google scholar*. Available at: https://scholar.google.com/ (Accessed: 07 January 2024).

Pak, A. and Paroubek, P. (no date) *Twitter as a corpus for sentiment analysis and opinion mining*, *ACL Anthology*. Available at: https://aclanthology.org/L10-1263/ (Accessed: 07 January 2024).

Xuanhui Wang University of Illinois at Urbana-Champaign *et al.* (2009) *Mining broad latent query aspects from search sessions: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and data mining*, *ACM Conferences*. Available at: https://dl.acm.org/doi/10.1145/1557019.1557114 (Accessed: 07 January 2024).

Raj, P. (2021) *Get data from Twitter API in Python - Step by step guide*, *Data Science Parichay*. Available at: https://datascienceparichay.com/article/get-data-from-twitter-api-in-python-step-by-step-guide/ (Accessed: 07 January 2024).

Shousha, H.M. (2017) *Apache spark streaming tutorial: Identifying trending Twitter hashtags: Toptal®*, *Toptal Engineering Blog*. Available at: https://www.toptal.com/apache/apache-spark-streaming-twitter (Accessed: 07 January 2024).

*Internet Archive: Digital Library of Free & Borrowable Books, movies, Music & Wayback Machine*. Available at: https://archive.org/details/twitterstream?sort=-publicdate (Accessed: 07 January 2024).