**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Modules Title:** | Advanced Data Analytics  Big Data Storage and Processing |
| **Assessment Title:** | MSC\_DA\_BD\_ADAv6 |
| **Lecturers Name:** | *David McQuaid*  *Muhammad Iqbal* |
| **Student Full Name:** | Alaa Shammary |
| **Student Number:** | SBA23361 |
| **Assessment Due Date:** | *23/05/2025 11.59pm* |
| **Date of Submission:** | 23/05/2025 11.59pm |

**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. |

Contents

[Introduction 3](#_Toc198847497)

[Big Data Processing and Storage 3](#_Toc198847498)

[Data Sources and Initial Preparation 3](#_Toc198847499)

[Data Storage 4](#_Toc198847500)

[Data Processing Environment 4](#_Toc198847501)

[Merging Processed Datasets and Final Storage 5](#_Toc198847502)

[Exporting the Final Dataset to CSV 5](#_Toc198847503)

[Comparative Analysis 6](#_Toc198847504)

# Introduction

This project investigates how public sentiment expressed on Twitter interacts with daily stock market movements and whether the two sources of information can be fused to improve short horizon price forecasts. The study combines 10 000 stock tagged tweets and one year of daily price data for large companies. After preprocessing, each tweet is scored for polarity; the resulting sentiment time series are aggregated to the ticker day level and joined to the corresponding market data.

# Big Data Processing and Storage

This section outlines the integration and processing of stock market and tweet datasets using distributed big data tools, specifically Cassandra, MongoDB, and Apache Spark.

## Data Sources and Initial Preparation

Two main datasets were used:

Tweet Dataset (stocktweet.csv): 10,000 entries containing tweet ID, timestamp, stock ticker, and tweet content.

Stock Prices Dataset (5 CSVs for AAPL, AMZN, MSFT, NVDA, TSLA): Each containing daily prices with columns including date, open, high, low, close, adjusted close, and volume.

The datasets were read using Pandas and a ticker column was added to each stock data frame for identification during merging and processing.

In this project, I implemented a robust pipeline for storing and processing large scale financial and social media data using modern Big Data technologies, with a focus on Cassandra, MongoDB, and Apache Spark. The rationale for the architecture and tool selection was guided by the nature of the data, scalability requirements, and the objective of performing real time and batch analytical operations efficiently.

## Data Storage

Stock Market Data (Structured Data):

The historical stock prices for companies such as AAPL, TSLA, MSFT, AMZN, and NVDA were stored in Cassandra, a NoSQL, distributed and highly scalable database optimised for time series data. Cassandra's schema was designed with a composite primary key (ticker, date) to enable fast queries by stock symbol and date range.

Justification: Cassandra was chosen for its high write throughput, fault tolerance and excellent support for time-based queries, which is ideal for stock price data that grows continuously over time.

Tweet Data (Semi-structured Text Data):

Tweets related to the selected stock tickers were stored in MongoDB, a flexible document-oriented database. Each tweet was inserted as a JSON-like document into a collection.

Justification: MongoDB was selected for its natural fit with semi structured data such as tweets, and its dynamic schema enabled quick ingestion and retrieval of diverse text entries.

## Data Processing Environment

Apache Spark:

Data processing, transformation and feature engineering were conducted using Apache Spark, specifically through PySpark. Spark was configured to read stock data from Cassandra and tweet data from MongoDB via appropriate connectors.

Key processing activities:

* Handling missing values (e.g., lagged values in stock data).
* Generating lagged close prices and future price columns (e.g., close\_t\_plus\_1, close\_t\_plus\_3).
* Creating new features such as daily returns and moving averages using Spark SQL functions and windowing operations.
* Performing aggregations to understand stock activity over time per ticker.
* Sentiment analysis on tweet content using PySpark, where tweets were classified as positive, negative, or neutral based on keyword heuristics. This enabled sentiment-based correlation with market movements and trading signals.

Justification: Spark was used for its in-memory distributed computing capabilities, which allow for fast processing of large datasets across clusters. Spark's integration with both Cassandra and MongoDB made it a natural choice for centralising processing without data migration.

## Merging Processed Datasets and Final Storage

After performing individual preprocessing steps on both datasets: tweets and stock prices using Apache Spark, the two data sets were joined to create a unified dataset. This merge was based on the stock ticker and the date of the stock market activity aligned with the tweet timestamp truncated to the same day for consistency.

The merging process involved the following Spark operations:

* Timestamp normalisation: Tweet timestamps were converted to date format using Spark SQL functions to enable joining on the same date field with the stock data.
* Inner join between the tweets and stock data based on the ticker and date columns.
* Selective filtering to retain relevant columns such as sentiment score, stock returns, moving averages, and future price indicators.

Storing the Merged Dataset in Cassandra

Once the unified dataset was created, it was stored back to Cassandra for durable and distributed storage. This step used the Spark-Cassandra connector, which allows writing a Spark data frame directly into a Cassandra table with appropriate schema mapping.

Justification:

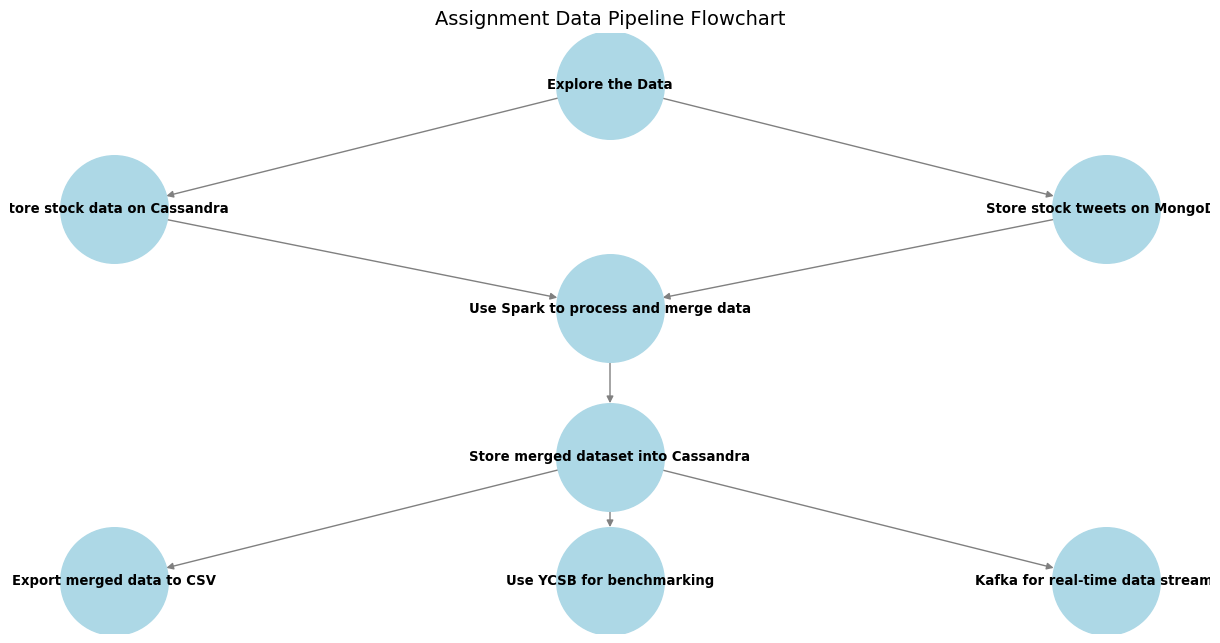
Cassandra was used again at this stage due to its low latency support for both read and write operations, and its ability to efficiently store the resulting time series data by composite keys (ticker, date).

Storing the enriched dataset (with both market and sentiment signals) in Cassandra enables downstream applications such as dashboards, alerts, or model training systems to access cleaned, feature rich data quickly.

## Exporting the Final Dataset to CSV

For advanced analysis using machine learning models built in traditional tools (e.g., Pandas, scikit-learn), the merged dataset was exported as a CSV file using PySpark’s.write.csv() method.

The following flowchart illustrates the sequence of steps followed in the Big Data pipeline for this project, from initial data exploration through processing, storage, export, and benchmarking.



## Comparative Analysis

To benchmark the performance of a relational database (MySQL) against a NoSQL database (MongoDB), the YCSB tool was used with Workload A, which represents a balanced mix of 50% reads and 50% updates. The results indicate that MongoDB significantly outperformed MySQL across all key metrics. MongoDB achieved a throughput of 254.12 operations per second, compared to 73.60 ops/sec for MySQL. Additionally, MongoDB demonstrated lower latencies in both read and update operations, with an average read latency of 853.42 µs versus 1508.91 µs in MySQL, and an average update latency of 6924.21 µs compared to 12865.35 µs. The 95th and 99th percentile latencies for reads were also lower in MongoDB, reflecting more consistent performance under load. These results suggest that MongoDB handles mixed workloads more efficiently, likely due to its document-based architecture and more flexible write model.