CS 839 - NoSQL Systems
Building a User-Friendly NoSQL-Based Football Analytics Platform: A Use-Case Study

Jainav Sanghvi
IMT2020098
Bhulaxmi Yash Koushik
IMT2020033

Abstract—**This project presents a novel approach to development of a user-friendly football analytics platform using NoSQL technologies. We aim to solve the problem of efficiently processing and analyzing structured football data to provide valuable insights and data-driven analysis for teams, players, and fans. The project uses a three-step process: data ingestion, data processing, and data analysis. The data ingestion step involves extracting the data from a variety of CSV files. The data processing step involves cleaning, transforming, and aggregating the data to prepare it for analysis. The data analysis step involves using NoSQL technologies to query and analyze the data to answer questions about football performance, team statistics, and match outcomes using a user-friendly website.**

The project makes the following contributions:

1) **A novel approach to football data analysis using NoSQL technologies.**
2) **A reusable and user-friendly pipeline.**
3) **A user-friendly front-end for analyzing football data.**

The project findings show that NoSQL technologies can be used to effectively process and analyze large datasets of football data and other similar dataset. The study explores the key features, technologies used, challenges faced, and the potential impact of the platform on the football industry.

1. Problem Definition:

1.1. Problem:

In this project, we aim to solve the problem of efficiently processing and analyzing football data using structured data processing techniques and NoSQL systems. The Football dataset, which encompasses detailed information about players, teams, and matches, serves as the primary source of data for our analysis. The overall goal of this project is to gain valuable insights into player performance, team statistics, and match outcomes through comprehensive data processing and analytics.

1.2. Background:

Football is a popular sport with a large fan base. There is a growing demand for data-driven insights into football performance, team statistics, and match outcomes. However, the lack of a scalable and efficient way to process and analyze large datasets of football data has limited the ability of researchers and analysts to answer these questions.

1.3. NoSQL Systems Perspective:

The main challenge lies in handling this complex and interconnected data in a scalable and flexible manner. From a NoSQL systems perspective, this problem is particularly interesting due to the nature and volume of the football dataset. By leveraging NoSQL systems, we address the challenge of handling complex and interconnected data in a scalable and flexible manner. It requires effective data modeling, querying, and processing techniques to extract meaningful information. Our main goal is to provide a platform that enables users to easily access and analyze football data while facilitating the creation of reusable and user-friendly data processing pipelines.

Traditional relational database systems may face challenges in handling the massive amounts of structured data present in the football domain. NoSQL systems, on the other hand, offer scalability, flexibility, and efficient processing capabilities for such large datasets. By utilizing NoSQL technologies, we can overcome the limitations of relational databases and effectively process and analyze the football data.

2. Approach:

In this project, we have adopted a comprehensive approach to solve the problem of processing and analyzing football data using structured data processing techniques and NoSQL systems. Our approach involves the utilization of specific algorithms, tools, and models, which are justified based on their suitability for the problem at hand. Additionally, we emphasize the simplicity and user-friendliness of our approach by building reusable pipelines.

2.1. NoSQL Technologies:

We leverage NoSQL technologies to handle the complex and interconnected nature of football data. Specifically, we use Flask, PyHive, and Hive for the backend.
Apace Hive and HiveQL: HiveQL, a SQL-like query language for Hive, is utilized to interact with the NoSQL database (Hive) and perform data analysis. We leverage the power of HiveQL to execute queries that calculate player statistics, team performance metrics, and predict match outcomes. Hive's ability to process large datasets in a distributed environment makes it a suitable choice for analyzing the football data.
Flask: Flask is a lightweight web framework that allows us to handle API requests and interact with the database. PyHive provides a Python interface for Hive, a data warehouse infrastructure, enabling efficient data processing and querying.
Justifications for Tool Selection: Flask is selected as the backend framework for its lightweight nature and simplicity in building web applications. It allows us to define routes and endpoints to handle requests from the frontend and connect with the Hive database via PyHive. Flask's flexibility enables us to establish a robust API layer that facilitates communication between the frontend and backend components.
We choose Hive as our NoSQL technology due to its integration with Hadoop, its SQL-like query language (HiveQL), and its ability to handle large-scale data processing. Hive provides a familiar SQL interface that simplifies the querying and analysis of structured data stored in distributed storage systems. Additionally, Hive's scalability and fault-tolerance capabilities align well with the requirements of handling extensive football datasets.

2.2. Dataset, Data Modeling and Storage:

The primary dataset used in this project is the Football dataset, which contains comprehensive information about football players, teams, and matches. This dataset serves as the foundation for our analysis and insights generation. It provides details such as player attributes, team performance metrics, and match outcomes. The dataset's richness and breadth make it a valuable resource for studying player performance, team dynamics, and match predictions.
We design a suitable data model to represent the football data in a structured manner. The data is stored in a NoSQL database HIVE, taking advantage of its flexible schema and scalability. By organizing the data into appropriate collections or tables, we ensure efficient retrieval and processing of information.

2.3. With and without Partition Approach:

To further optimize the data analysis process, we employed partitioning in Hive by partitioning the data based on the 'tournament name' column. This technique significantly reduced the runtime of queries, resulting in improved performance.

Partitioning in Hive involves dividing the data into smaller, more manageable segments based on specific criteria. In this case, partitioning was done based on the 'tournament name'. The advantage of using partitioning is that it allows Hive to operate on a subset of data, rather than scanning the entire dataset for each query. By narrowing down the search space to a specific partition, the query execution time is greatly reduced.

The key difference between using partitioning and not using it is the level of data processing required. Without partitioning, Hive needs to scan the entire dataset for each query, which can be time-consuming and resource-intensive. However, with partitioning, Hive only needs to access the relevant partition(s) that satisfy the query condition, resulting in faster query execution.

By leveraging the power of partitioning in Hive, I was able to achieve significant performance improvements in data analysis. The ability to process large datasets in a distributed environment combined with the efficiency of partitioning makes Hive a valuable tool for analyzing football data and extracting meaningful insights efficiently.

2.4. Front-End Interface with React:

The front-end of our platform is built using React, a popular JavaScript library known for its efficiency and flexibility in creating user interfaces. React enables us to develop a dynamic and interactive interface that enhances the user experience by providing real-time updates and seamless navigation through different views.

In our project, the front-end interface mainly comprises three pages: player details and stats, team stats for the FIFA World Cup, and FIFA tournament stats for a specific year. Each page offers users specific functionalities and insights into football data. Through the React components, users can easily interact with the platform, selecting parameters such as country, player name, team, timeline, and desired statistics.

When a user selects specific parameters, the React components capture these selections and send them to the backend Hive database through the Flask API. This allows for efficient processing and analysis of the data based on the user's input. The frontend interface provides an intuitive and visually appealing experience, ensuring that users can navigate effortlessly and access relevant information.

The React components handle user input, form validation, and state management, making the front-end highly responsive. As users interact with the interface, React components update the application state and trigger the necessary API requests to the backend. This real-time communication between the frontend and backend enables a seamless flow of data and ensures that users receive up-to-date information and analysis.

2.5. Data Pipelines:

One of the key objectives of our approach is to develop user-friendly and reusable data processing pipelines. To achieve this, we leverage various tools and techniques that simplify the pipeline development and enhance the overall usability of the platform.

Our approach focuses on developing reusable and user-friendly data processing pipelines, which are instrumental in extracting meaningful insights from the football dataset. The pipeline consists of three essential steps: data ingestion, data cleaning and pre-processing, and data analysis and visualization.

1) **Data Ingestion:** The first step in our pipeline is data ingestion, where we import the football dataset into Hadoop and store it in HDFS (Hadoop Distributed File System). Hadoop provides a scalable and distributed storage system that can handle large volumes of structured data efficiently. By leveraging HDFS, we ensure that the dataset is readily accessible for further processing and analysis.
2) **Data Cleaning and Pre-processing:** Once the data is ingested into Hadoop, the next step is to clean and pre-process it. Data cleaning involves identifying and handling issues such as missing values, duplicates, and inconsistencies. Pre-processing tasks may include data transformation, standardization, and feature engineering to ensure the data is in a consistent and suitable format for analysis.
   By addressing data quality and consistency concerns, we ensure that the subsequent analysis is based on reliable and accurate data. This step plays a crucial role in improving the overall data quality and eliminating potential biases or errors that could impact the analysis results.
   In our case, our dataset was well structured, and hence we did not require any data cleaning or preprocessing techniques. As the data was already cleaned and highly structured, we directly loaded it into the Hive tables without the need for additional preprocessing steps. This allowed us to proceed with the subsequent stages of our analysis pipeline more efficiently, focusing on the analysis and insights generation tasks.
3) **Data Analysis and Visualization:** After the data cleaning and pre-processing step, we proceed with data analysis and visualization. Apache Hive, a SQL-like query language, is employed as a powerful tool for querying and analyzing the structured data stored in Hadoop. Hive enables us to perform various analyses, such as calculating player statistics, team statistics, and predicting match outcomes.

2.6. User-friendliness and Reusability

Our approach aims to simplify the development of reusable and user-friendly data processing pipelines. By leveraging tools like Flask and React, we create a seamless interface between the frontend and backend components. The interface provides intuitive controls and visualizations to present the analyzed data in a user-friendly format, allowing users to interpret and explore the results effortlessly.

By structuring the pipeline into these three distinct steps – data ingestion, data cleaning and pre-processing, and data analysis and visualization – we ensure that the pipeline is modular, reusable, and user-friendly. Users can easily understand and adapt the pipeline to other datasets or future use-cases by following the same sequence of steps. This promotes efficiency, consistency, and scalability in the data processing and analysis tasks, enhancing the overall usability and effectiveness of the platform.

By incorporating these practices, our approach simplifies the development and deployment of data processing pipelines. The user-friendly nature of the frontend interface, coupled with the reusable pipelines, allows users to easily load, process, and analyze data without requiring extensive knowledge of the underlying technologies. This approach promotes usability and ensures that users can efficiently derive insights from the football dataset while maintaining the flexibility to adapt the pipelines for future use-cases.

2.7. Conclusion :

By employing this approach, we create a football analytics platform that enables users to efficiently process and analyze structured football data. The integration of NoSQL technologies, a user-friendly interface, and reusable data processing pipelines ensures a seamless experience for users, empowering them to gain valuable insights into team performance, player statistics, and match outcomes.

3. System Demo and Evaluation:

For Player Details and Statistics:
Time taken to run: 89secs

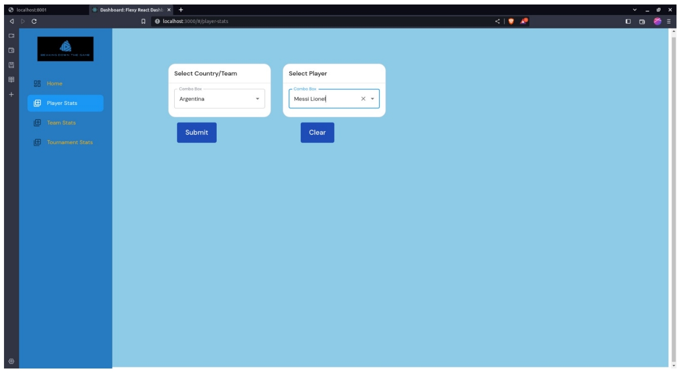
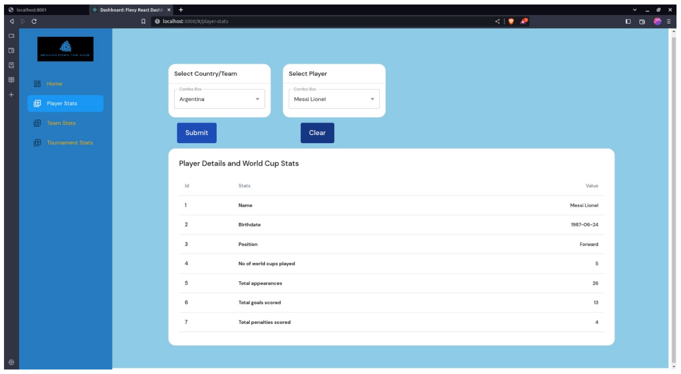


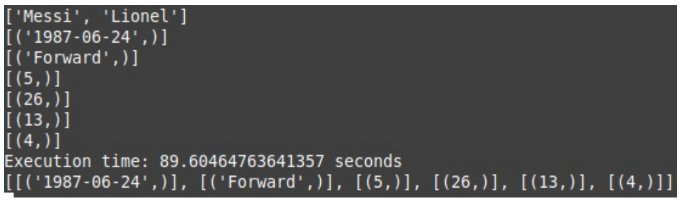
Figure 1.



Player Details and World Cup Stats

Id	Stats	Value
1	Name	Messi Lionel
2	Birthdate	1987-06-24
3	Position	Forward
4	No of world cups played	5
5	Total appearances	26
6	Total goals scored	13
7	Total penalties scored	4

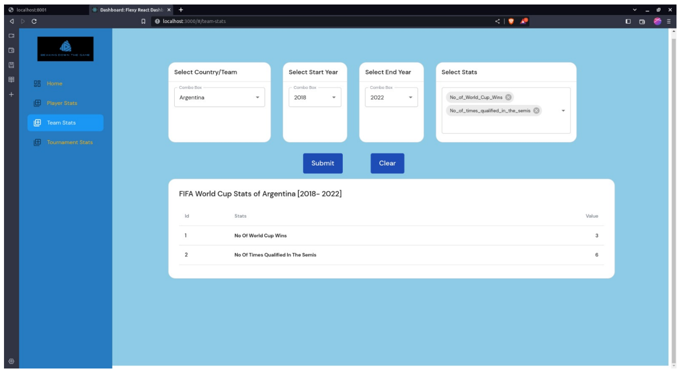
Figure 2.



```
['Messi', 'Lionel']
[('1987-06-24',)]
[('Forward',)]
[(5,)]
[(26,)]
[(13,)]
[(4,)]
Execution time: 89.60464763641357 seconds
[[('1987-06-24',)], [('Forward',)], [(5,)], [(26,)], [(13,)], [(4,)]]
```

Figure 3.

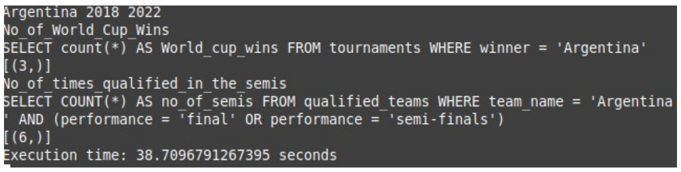
Similarly the results for Team Statistics and Tournament Stats:
Time taken to run team statistics: 38.5 secs
Time taken to run tournament statistics: 0.82 secs



FIFA World Cup Stats of Argentina [2018- 2022]

Id	Stats	Value
1	No Of World Cup Wins	3
2	No Of Times Qualified In The Semis	6

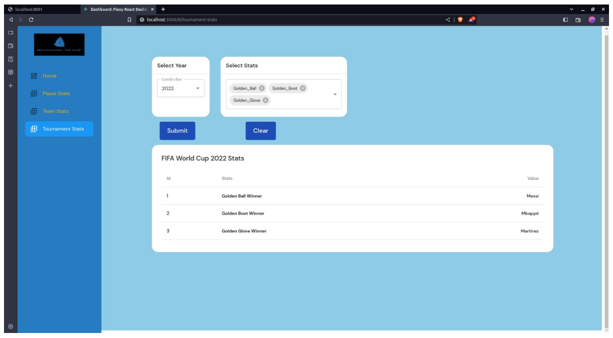
Figure 4.



```
Argentina 2018 2022
No_of_World_Cup_Wins
SELECT count(*) AS World_cup_wins FROM tournaments WHERE winner = 'Argentina'
[(3,)]
No_of_times_qualified_in_the_semis
SELECT COUNT(*) AS no_of_semis FROM qualified_teams WHERE team_name = 'Argentina
' AND (performance = 'final' OR performance = 'semi-finals')
[(6,)]
Execution time: 38.7096791267395 seconds
```

Figure 5.

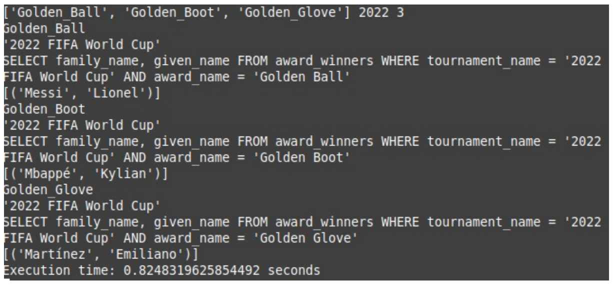
With partitions approach, the runtime reduced drastically. It gave results almost in real time



FIFA World Cup 2022 Stats

Id	Stats	Value
1	Golden Ball Winner	Messi
2	Golden Boot Winner	Mbappé
3	Golden Glove Winner	Martínez

Figure 6.



```
['Golden_Ball', 'Golden_Boot', 'Golden_Glove'] 2022 3
Golden_Ball
'2022 FIFA World Cup'
SELECT family_name, given_name FROM award_winners WHERE tournament_name = '2022
FIFA World Cup' AND award_name = 'Golden Ball'
[('Messi', 'Lionel')]
Golden_Boot
'2022 FIFA World Cup'
SELECT family_name, given_name FROM award_winners WHERE tournament_name = '2022
FIFA World Cup' AND award_name = 'Golden Boot'
[('Mbappé', 'Kylian')]
Golden_Glove
'2022 FIFA World Cup'
SELECT family_name, given_name FROM award_winners WHERE tournament_name = '2022
FIFA World Cup' AND award_name = 'Golden Glove'
[('Martínez', 'Emiliano')]
Execution time: 0.8248319625854492 seconds
```

Figure 7.

4. References:

1) React documentation
2) Flask documentation
3) React and flask connection blogs
4) HiveQL and pyhive tutorials and blogs