

Developing a Model that Enables Real Time Streaming of Transactions in Financial Institutions Case Study: Bank of Kigali PLC

Tuyisenge Marie Josee¹; Dr. Wilson Musoni² (PhD)

^{1,2}Masters of Science with Honors in Information Technology at University of Kigali, Rwanda

Publication Date: 2025/04/12

Abstract: Recent technological advancements, particularly in banking and financial services, have profoundly influenced the development and use of applications. The increasing demand for modern banking systems to address consumer expectations, speed up market entry, enhance user experiences, improve security, and optimize system efficiency has led to the adoption of new technologies. This trend has sparked the creation of numerous banking-specific modernization solutions. Within the interconnected banking, financial services, and insurance (BFSI) sectors, the need for effective enterprise application integration (EAI) is crucial for gaining both technological and business advantages. Real-time data streaming through technologies like Apache Kafka has become essential for enabling seamless communication between systems. Kafka, a widely used distributed streaming platform, surpasses traditional message queues in scalability, message retention, data replication, and processing order, making it an essential part of modern banking systems. Unlike batch processing, which processes large volumes of data at set intervals and can cause delays in transactions, streaming technologies eliminate lag by processing data continuously in real-time, significantly enhancing the speed and accuracy of financial operations. The shift from batch systems to real-time streaming allows BFSI applications to process transactions and settlements more quickly, reducing processing times and improving operational efficiency. Additionally, the use of Kafka enhances the security and reliability of data transfers, making it a valuable tool for modernizing banking infrastructures. Therefore, integrating real-time streaming technologies like Kafka not only improves system performance but also boosts security, scalability, and operational effectiveness, enabling financial institutions to better meet customer demands in an increasingly digital economy.

How to Cite: Tuyisenge Marie Josee; Dr. Wilson Musoni. (2025). Developing a Model that Enables Real Time Streaming of Transactions in Financial Institutions Case Study: Bank of Kigali PLC. *International Journal of Innovative Science and Research Technology*, 10(3), 2654-2664. <https://doi.org/10.38124/ijisrt/25mar1746>.

I. INTRODUCTION

This research explores the implementation of a real-time data processing model at the Bank of Kigali, focusing on enhancing operational efficiency and customer service. By leveraging advanced technologies such as Apache Kafka, Python, and the Avro schema, the study investigates a system that extracts, transforms, and loads (ETL) data into relational databases to ensure faster, accurate decision-making. Data is extracted from an Oracle database using Kafka, serialized with Avro, and transformed into a relational format using Python before being stored in an MSSQL database. The model aims to automate data processing, reduce human error, and enable real-time insights for improved customer relationship management, personalized marketing, and fraud detection. This research highlights the potential of big data technologies to streamline workflows and improve both business operations and customer satisfaction in the banking sector.

➤ Statement of the Problem

The Banking and Financial Services Industry (BFSI) has traditionally depended on batch processing for managing transactions, data processing, and settlements. However, this outdated method often results in delays, inefficiencies, and a lack of real-time data visibility, hindering financial institutions' ability to respond swiftly and maintain their competitiveness. With increasing competition from FinTech companies and startups, evolving business models, stricter regulatory requirements, and rapid technological advancements, BFSI institutions are under pressure to modernize their systems. Today's tech-savvy customers expect fast access to services, real-time transaction processing, and instant insights into their financial information—expectations that current batch systems cannot meet. Batch processing, which typically follows a Close of Business (COB) cycle, causes delays in addressing client issues such as mismatched balances or failed transactions, as it only processes information after the business day concludes. Additionally, the inability to easily exchange data across systems leads to operational bottlenecks and limits

innovation. To overcome these challenges, integrating a streaming platform into banking and financial systems is essential. The goal of this project is to develop and implement a reliable streaming system that enables real-time data processing, improves operational efficiency, ensures compliance with regulations, and enhances user experience by providing immediate access to financial services. By effectively adopting streaming technology, BFSI institutions will be able to transition from outdated batch processing systems to a more modern, agile, and responsive infrastructure, helping them stay competitive in the rapidly evolving financial sector

II. LITERATURE REVIEW

This chapter focuses on the study of real-time streaming of transactions at the Bank of Kigali, providing an in-depth review of key concepts, terms, and relevant literature related to the research.

A. Real-Time Streaming of Transactions

Real-time streaming plays a crucial role at the Bank of Kigali, particularly in the data management department, where it supports real-time data processing and analysis. Data from T24 Transact is captured and inserted into the F_Data_Events table for further processing.

B. T24 Transact

T24 Transact, developed by Temenos, is the bank’s core banking system that handles key banking operations like account management, deposits, loans, and payments. It supports real-time transaction processing, ensuring operational efficiency and compliance. The system is flexible and can be customized for various banking models while integrating easily with other financial services.

C. Staging Database (Oracle)

The staging database temporarily stores deserialized events before they undergo further processing.

D. MSSQL Database

Microsoft SQL Server (MSSQL) is an RDBMS used to manage large volumes of structured data. It supports complex queries and real-time data processing, ensuring data integrity and high availability. It also offers tools for business intelligence and data security.

E. Operational Data Store (ODS)

An ODS is used to store real-time operational data from various sources. It serves as a bridge between transactional systems and data warehouses, providing up-to-date data for daily operations and decision-making.

F. Snapshot Data Store (SDS)

An SDS stores snapshots of data at specific times, enabling historical analysis and trend tracking. It is useful for auditing, financial reporting, and understanding data changes over time.

G. Theoretical Review

➤ Kafka

Kafka is a distributed event streaming platform that handles high-throughput, low-latency messaging. It collects, stores, and processes large amounts of real-time data, ensuring durability and fault tolerance. Kafka’s publish-subscribe architecture makes it suitable for real-time analytics and scalable data pipelines, supporting event-driven applications and real-time data streaming.

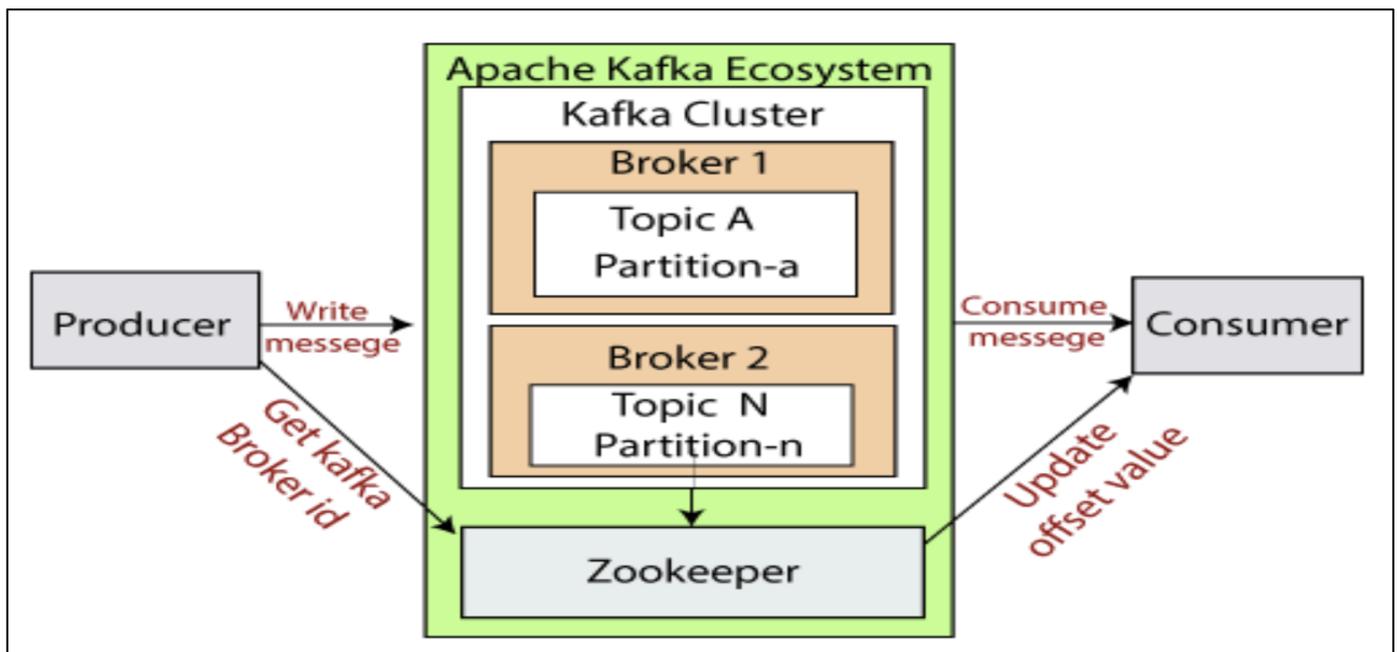


Fig 1: Kafka Ecosystem
Source: Own Capture with Lightshot, 2025

➤ *Avro Schema*

Avro is a data serialization format used to encode events before they are stored in Kafka. It applies the Avro schema standard for data serialization, which involves encoding and decoding data in a structured way. By using a defined schema, Avro ensures that the serialized events are both compact and efficient. It enables data to be serialized into a binary format, which boosts performance and reduces storage

requirements. The schema clearly defines the data structure, ensuring consistent processing and interpretation of events. This structure enhances interoperability and simplifies integration with various systems utilizing Kafka streams. Additionally, Avro supports schema evolution, allowing changes to the data format without disrupting existing consumers. In summary, the Avro schema makes event-driven architectures more efficient, reliable, and flexible for data serialization.

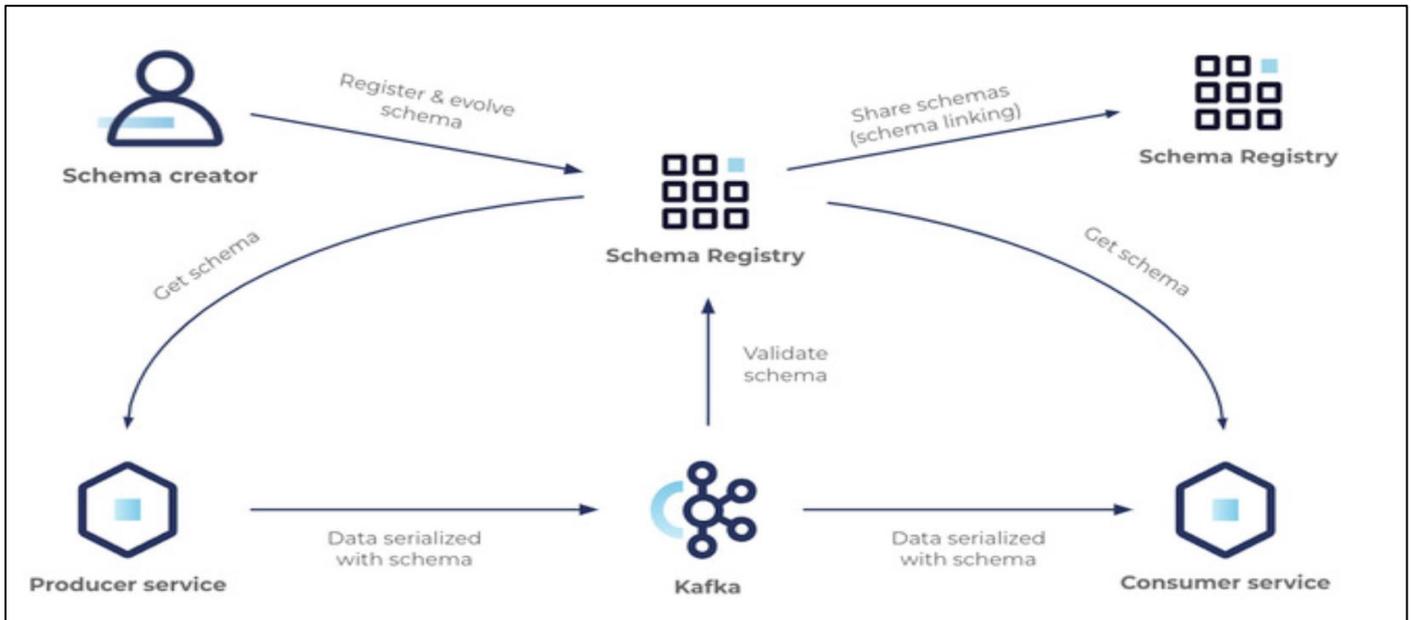


Fig 2: Avro Schema Figure
Source: Own Drawing, 2025

➤ *Zookeeper*

Zookeeper is a centralized system that provides services for group management, distributed synchronization, naming, and maintaining configuration data. These services are essential for distributed applications in various ways. However, addressing inevitable flaws and race conditions often requires significant effort each time these services are deployed. Due to the complexity of implementing these services, applications often take shortcuts initially, making them susceptible to changes and difficult to maintain. Even when these services are implemented correctly, different implementations can lead to increased management complexity when the applications are deployed.

➤ *Python*

Python 3 is a flexible and widely used programming language for tasks involving data loading, processing, and transformation. It supports multiple data sources such as CSV files, SQL databases, and APIs through libraries like Pandas, SQLAlchemy, and Requests. NumPy handles numerical computations efficiently, while Pandas is excellent for data cleaning, filtering, and aggregation. PySpark enables large-scale data processing in distributed systems, and the re module helps with pattern extraction and text cleaning.

Python also allows transformed data to be saved back into databases or files (CSV, JSON, Excel) using functions like `to_sql()` and `to_csv()`. Additionally, PySpark can store data on distributed file systems such as HDFS or S3, as well as in cloud storage. With its wide range of libraries, Python is well-suited for building effective ETL (Extract, Transform, Load) pipelines, ensuring smooth processing, transformation, and integration of data. Python 3 is therefore crucial for efficient workflows in data engineering, analysis, and machine learning.

➤ *Linux Server*

A Linux server is a computer running the Linux operating system that provides network services to other systems. It is commonly used for file sharing, database management, and web hosting due to its performance, security, and stability. As a free and open-source operating system, Linux allows for customization. Servers typically use command-line interfaces to improve efficiency. Popular Linux distributions for servers include CentOS, Ubuntu, and Debian. Linux is widely adopted in business environments because of its cost-effectiveness and scalability, supporting services like email hosting, network management, and web hosting.



Fig 3: Linux Server

Source: Own Capture with Lightshot, 2025

➤ *Window Server*

Windows Server is an operating system developed by Microsoft to manage network resources and provide services to other computers. It supports functions like database management, web hosting, and file sharing. Unlike many other server operating systems, Windows Server offers a user-friendly graphical user interface (GUI), making it easier

to use. It includes Active Directory for managing users and network resources. The system is secured by technologies such as firewalls and Windows Defender. Windows Server also supports virtualization through tools like Hyper-V, making it ideal for running critical business applications in enterprise environments. Remote management is facilitated through features like Remote Desktop and PowerShell.



Fig 4: Window Server

Source: Own Capture with Lightshot, 2025

III. METHODOLOGY AND RESEARCH

This chapter addresses the solutions to the research problem, covering aspects such as research design, population and sample selection, data collection methods, data analysis, validity, reliability, and ethical considerations.

➤ *Research Design*

The researcher employed a combination of techniques in the research design, including questionnaires, interviews, literature reviews, and data collection methods. The study explored the relationship between independent and dependent variables through a predictive approach, incorporating an intervening variable to summarize these relationships. Each sample member was assessed based on multiple variables.

Both qualitative and quantitative methods were utilized in the study.

As described by Rhodes (2014), the qualitative approach aimed to provide a deep, descriptive understanding of the phenomenon, typically through interviews, open-ended questions, or focus groups. Meanwhile, the quantitative approach was selected to examine a larger sample and identify general patterns through statistical analysis, enabling the data to be quantified numerically. This approach allows for identifying trends across a larger population.

➤ *Area of Study*

The research was conducted at Bank of Kigali PLC, located in Kigali city, Rwanda, specifically in the Nyarugenge district and Nyarugenge Sector.



Fig 5: Bank of Kigali
Source: Own Capture with Lightshot, 2025

IV. CONCEPT FRAMEWORK (INDEPENDENT VARIABLE AND DEPENDENT VARIABLE)

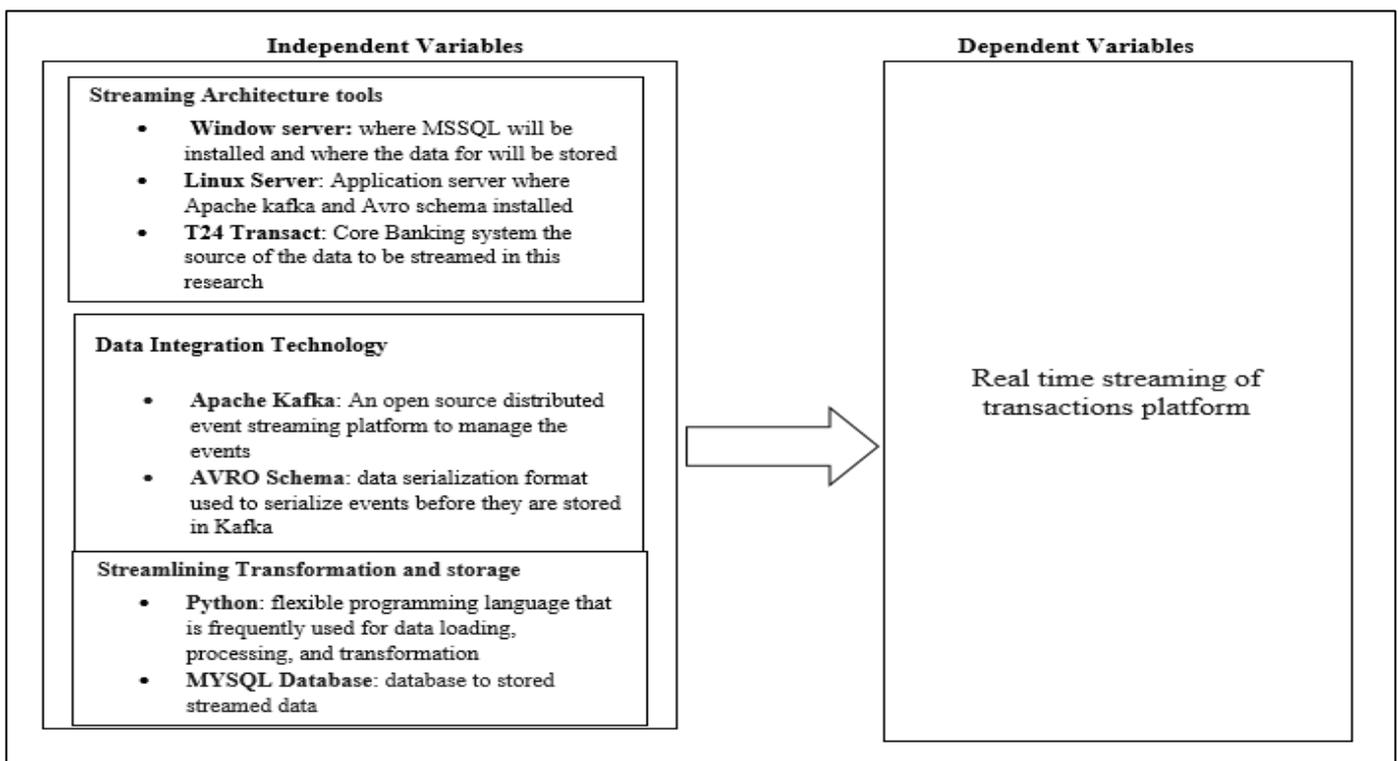


Fig 6: Conceptual Framework

V. DATA ANALYSIS, PRESENTATION, AND INTERPRETATION

➤ *Design System*

The system design follows a three-step process: data extraction, transformation, and storage. Each stage incorporates various tools and technologies to facilitate the seamless movement of data from the source to its destination. In the extraction step, data is collected from multiple sources. The transformation phase involves processing the data and converting it into a suitable format. In the final step, the data is stored in the appropriate database or storage system, ensuring both accuracy and efficiency.

The flow begins with data stored in an Oracle database, which Kafka Connect extracts and serializes using the Avro schema. This data is then placed into Kafka topics, ready for further processing. Python handles the transformation of this data into a relational format, which is subsequently stored in an MSSQL database. The process of converting data from XML to a relational structure takes just 5 seconds per record, ensuring a quick and efficient data pipeline. By leveraging the Avro schema and Kafka for scalable and reliable data movement, along with Python for data transformation, the system effectively bridges the gap between different data formats and databases.

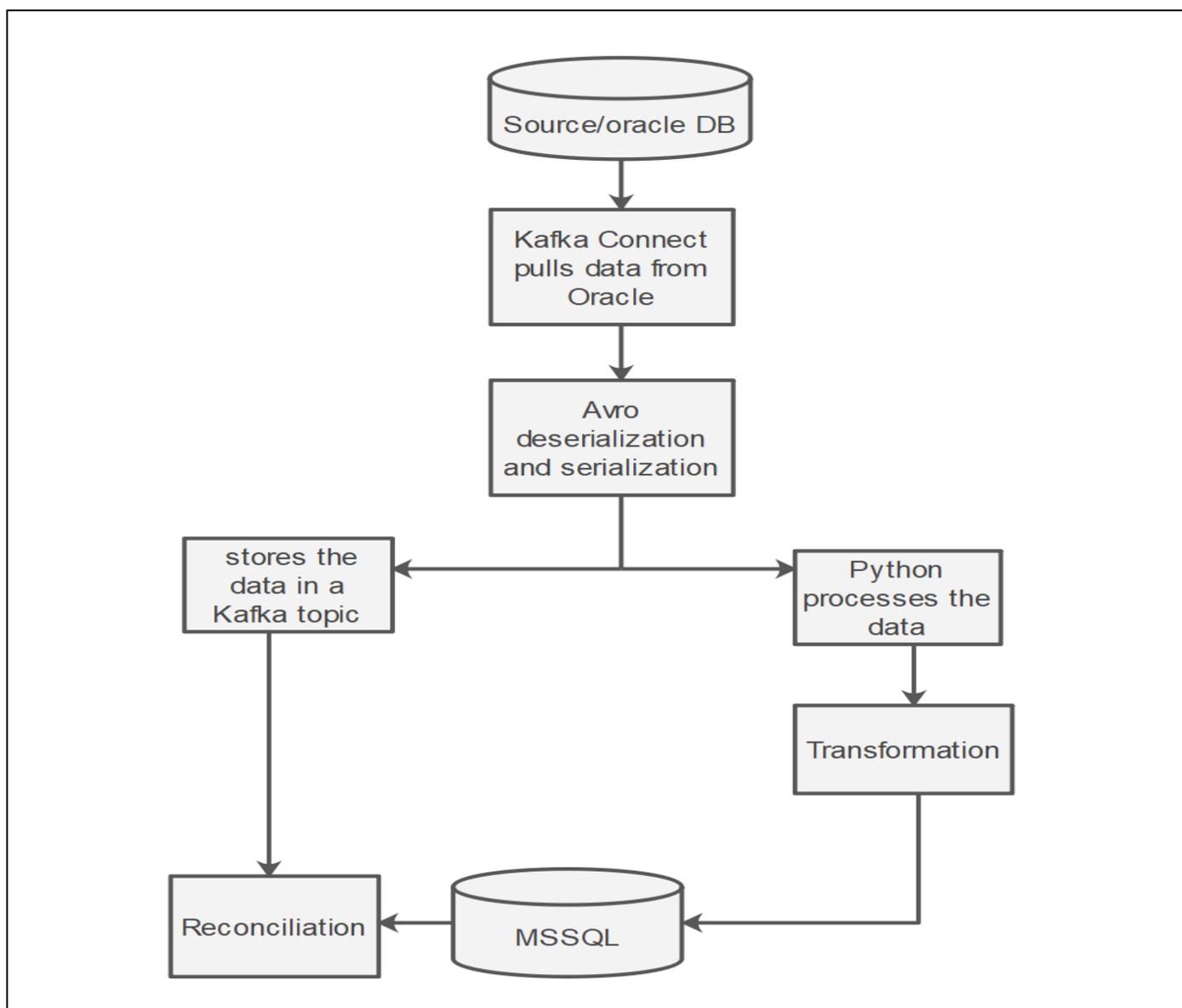


Fig 7: Flow Chart
Source: Own Drawing with Lightshot, 2025

➤ Process Diagram

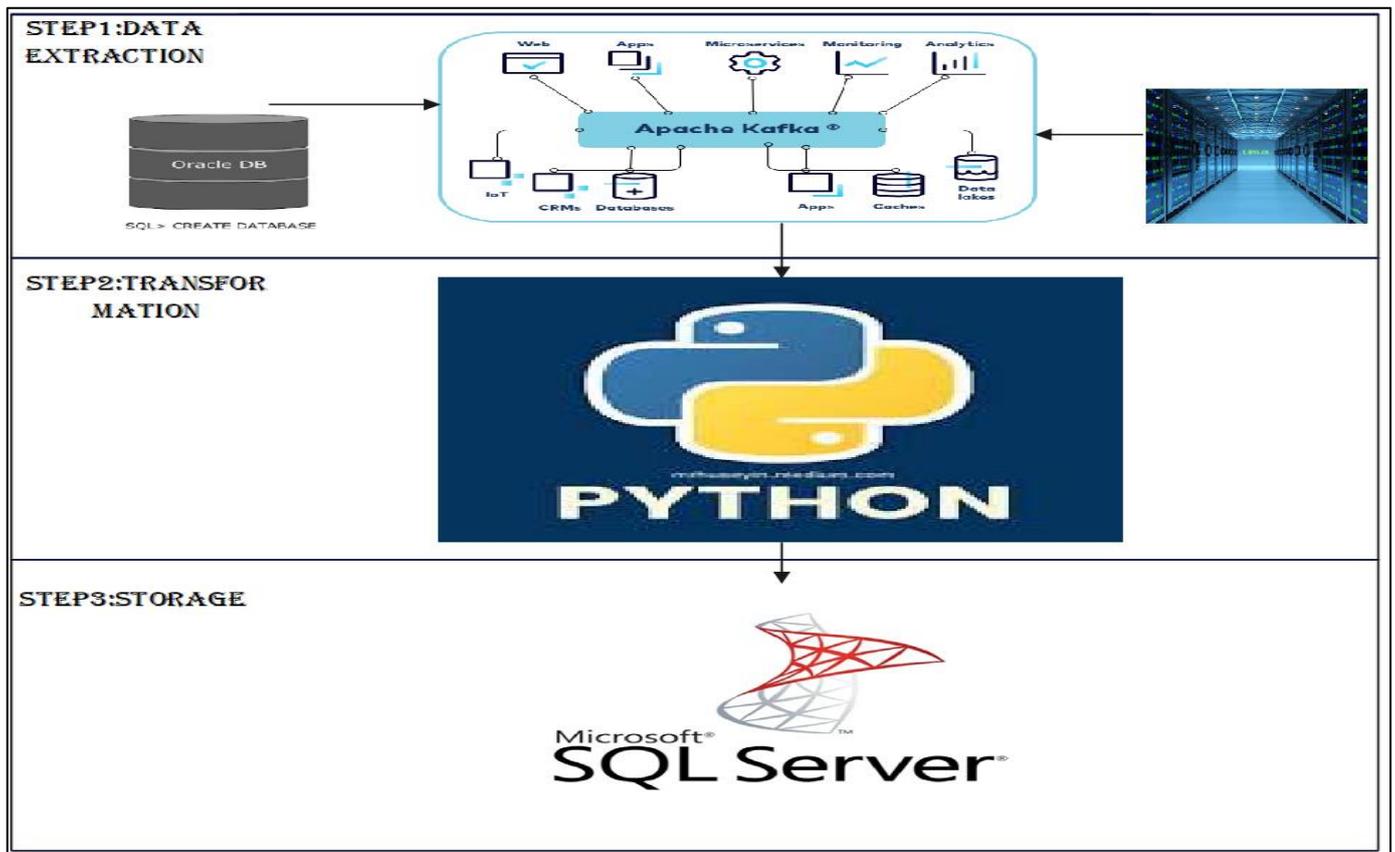


Fig 8: Process Diagram
Source: Own Drawing with Lightshot, 2025

Online streaming is a blend of both art and science, with big data strategies often requiring creative solutions to overcome unique challenges. In this case, the combination of Python scripting, Apache Kafka, and the Avro schema proved to be a highly effective approach. Python facilitated efficient data processing and automation, while Apache Kafka enabled real-time streaming and scalable data transmission. The Avro schema, which integrates seamlessly with Kafka, ensured consistent data serialization and schema validation, allowing for smooth interaction between various systems.

VI. CUSTOMER

This table holds all client information used by the bank, including sensitive data such as names, addresses, identification numbers, and customer type (individual or business). It plays a vital role in ensuring effective customer relationship management (CRM). Key fields in the table include customer ID, name, contact information, customer type, and country, among others.

The screenshot shows a SQL query results window. The query is 'SELECT * FROM T24.FBNK.CUSTOMER'. The results are displayed in a table with the following columns: RECID, XMLRECORD, MNEMONIC, CUSTOMER_TYPE, and SECTOR. The 'CUSTOMER' table name in the query is highlighted with a red box.

	RECID	XMLRECORD	MNEMONIC	CUSTOMER_TYPE	SECTOR
1	1001632	[XML]	BK1001632	[NULL]	2,001
2	1001631	[XML]	BK1001631	ACTIVE	2,000
3	1001634	[XML]	BK1001634	[NULL]	2,001
4	1001633	[XML]	BK1001633	[NULL]	1,000
5	1001630	[XML]	BK1001630	ACTIVE	1,001
6	1001630	[XML]	BK1001630	[NULL]	1,000

Fig 9: Customer Field

➤ *Account*

The accounts table contains details about the various accounts customers maintain with the bank, including savings, checking, and fixed deposit accounts. It tracks account balances, types, and associated services such as interest rates or fees.

➤ *Key Fields:*

Account Number, Customer ID, account type, balance, currency type.

Grid	RECID	XMLRECORD	CO_CODE	CATEGORY	CUSTOMER	DATE_LAST_CR_CUST	DATE_LAST_DR
1	100001448387	[XML]	RW0010015	1,001	1,017,305	20,250,114	
2	100028743647	[XML]	RW0010005	1,011	1,185,639	[NULL]	
3	100000615287	[XML]	RW0010002	1,003	1,005,384	20,241,128	
4	100028682168	[XML]	RW0010053	1,001	1,373,752	20,220,104	
5	100028985977	[XML]	RW0010004	1,001	1,375,120	20,220,228	

Fig 10: Account Field
Source: Own Capture with Lightshot

Conduktor is a powerful and user-friendly tool designed to simplify the development and management of Apache Kafka. Its primary purpose is to enhance the ease of

maintaining, expanding, and monitoring Kafka clusters login to manager kafka platform where you can add topics and monitor how it is behaving.

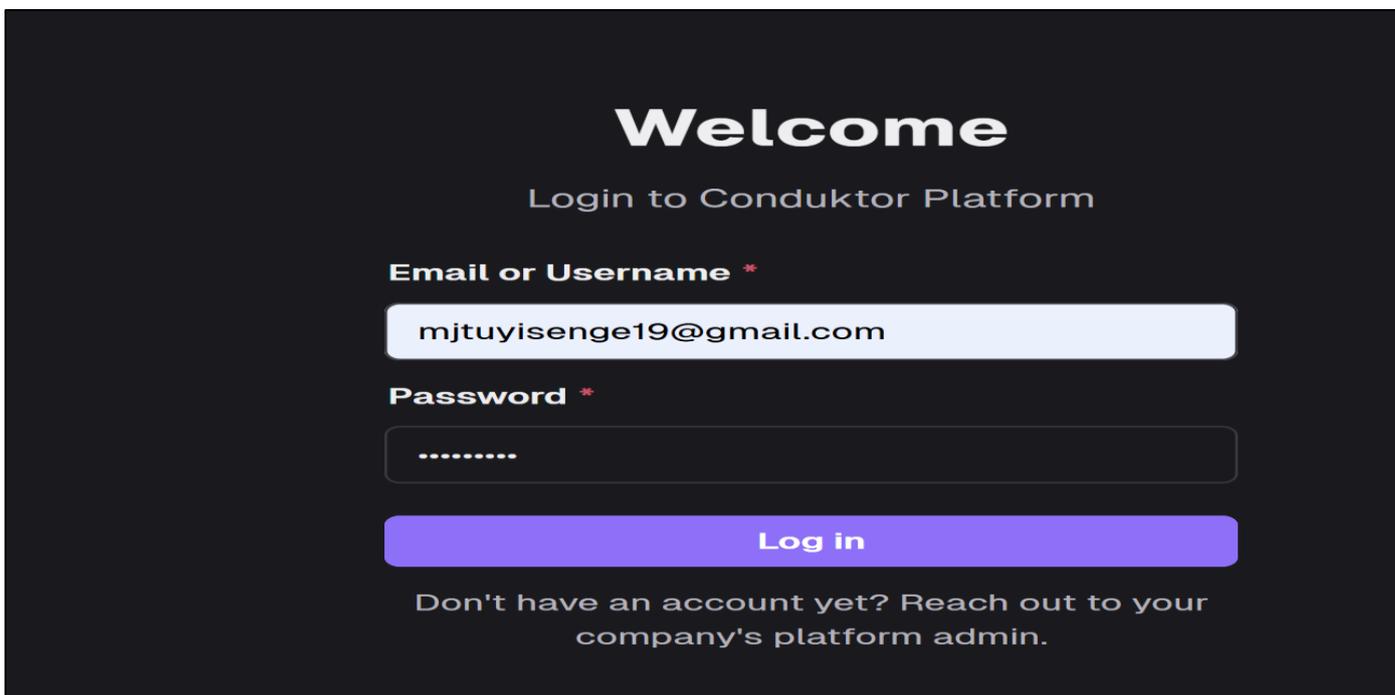


Fig 11: Login Page
Source: Own Capture with Lightshot

After login you can add the bootstrap server and test a connection if it is successful you can continue with the monitoring the data.

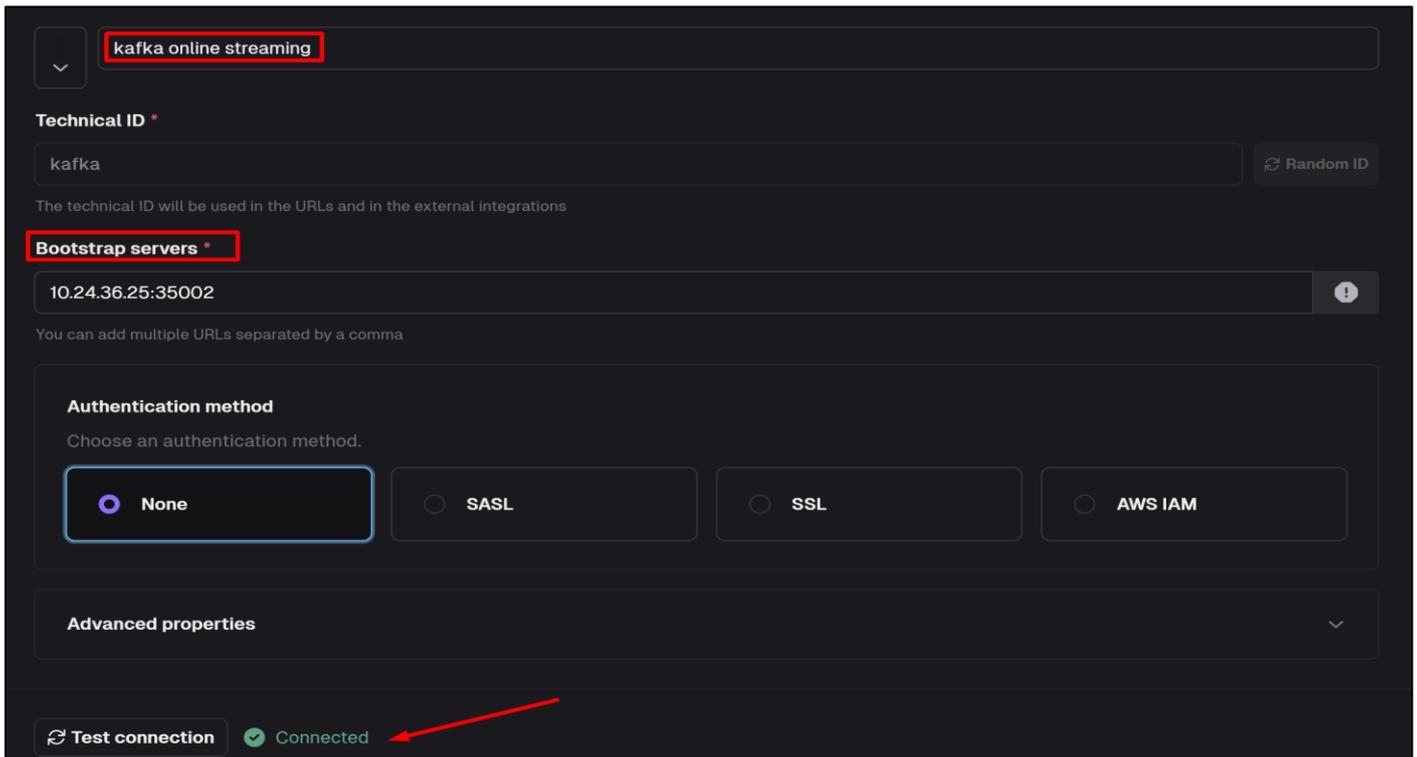


Fig 12: Schema Connection
 Source: Own capture with lightshot

Below is the list of topics, but I will focus on eot which hold the data that need to be transformed in our project of online streaming.

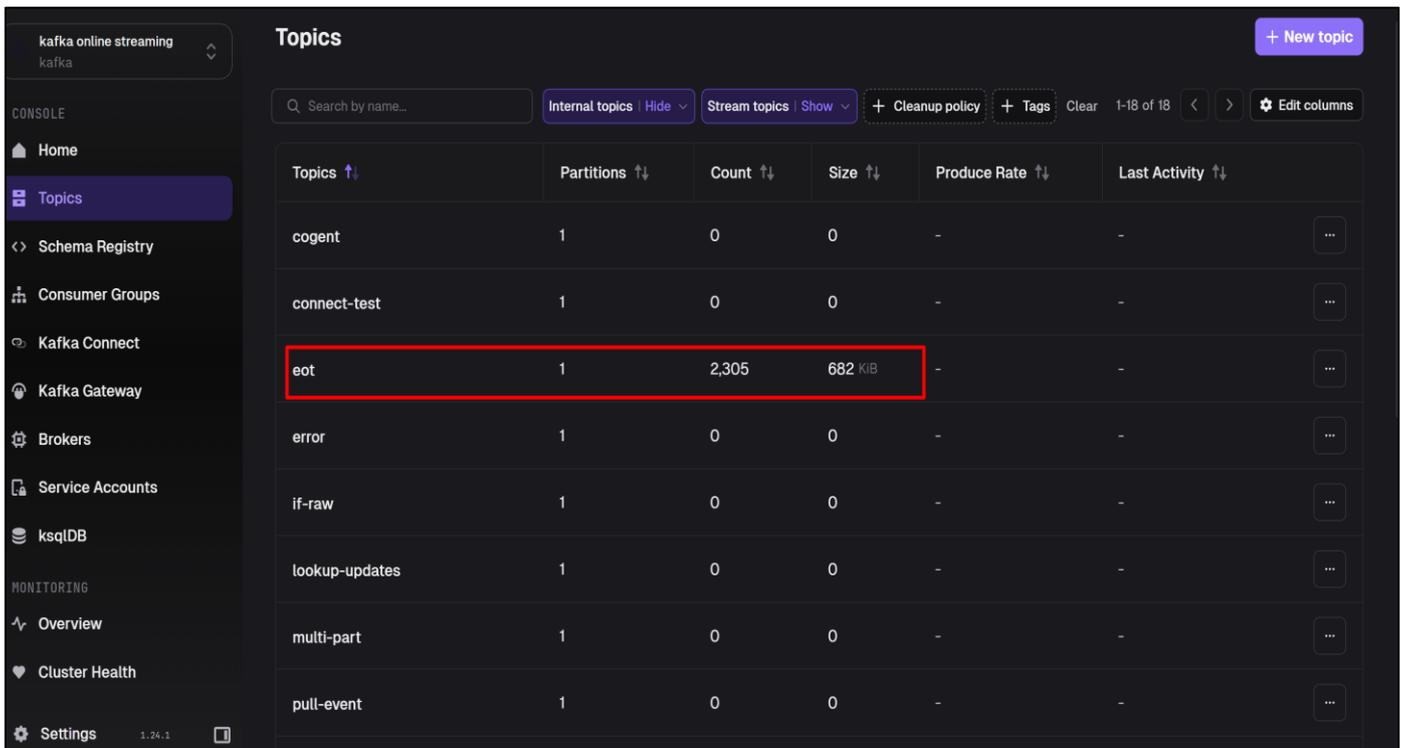


Fig 13: Kafka Topics

➤ *Offset*

Offset Explorer is a data monitoring tool that helps track how data moves from the source to the destination. Similar to

Conduktor, it provides deeper insights into the data within topics, whereas Conduktor focuses on maintaining the overall health of the data loading process.

Partition	Offset	Key	Value	Timestamp
0	21758892	55363630363332365F386635353266...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758893	55363630363332365F356664626333...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758894	55363630363332365F336662633036...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758895	55363630363332365F643238393836...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758896	55363630363332365F343631626436...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758897	55363630363332365F663438663430...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758898	55363630363332365F633335613362...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758899	55363630363332365F376435646235...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758900	55363630363332365F623637383664...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758901	55363630363332365F343937366439...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758902	55363630363332365F373065643931...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758903	55363630363332365F32623316638...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758904	55363630363332365F346363616637...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758905	55363630363332365F623239646461...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758906	55363630363332365F643534313234...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758907	55363630363332365F613764326137...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758908	55363630363332365F343931363361...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758909	55363630363332365F666239346630...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758910	55363630363332365F373430666433...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758911	55363630363332365F343038316438...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758912	55363630363332365F613263306661...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758913	55363630363332365F346236393535...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758914	55363630363332365F626463653637...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758915	55363630363332365F613331323965...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758916	55363630363332365F383735313363...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758917	55363630363332365F393865643332...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758918	55363630363332365F643932353835...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758919	55363630363332365F366433626331...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758920	55363630363332365F333366353565...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758921	55363630363332365F356134373039...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758922	55363630363332365F363861633834...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758923	55363630363332365F383130333738...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758924	55363630363332365F383362363461...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758925	55363630363332365F623235306265...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758926	55363630363332365F616561376631...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758927	55363630363332365F356666326233...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758928	55363630363332365F656334366164...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597
0	21758929	55363630363332365F3866563353364...	00000005CD6455363630363332365...	2025-02-14 15:00:48.597

Fig 14: Data Processing

VII. CONCLUSIONS

In conclusion, the online streaming model has proven to be an effective and reliable solution for capturing and transforming data into a relational database. This research has deepened my understanding of big data and its potential societal impact. The process begins with transaction initiation, where Kafka immediately captures the data in real time, sending it to Python for transformation. The transformed data is then automatically loaded into the relational database, eliminating manual intervention and minimizing human error. The transformation, completed in approximately 5 seconds, ensures timely updates, which is vital for enhancing customer service, particularly at the Bank of Kigali.

Using this model, financial institutions can offer quicker, more accurate responses to customer needs, improving satisfaction and enabling more informed decision-making. The system also reduces operational costs and streamlines processes. Ultimately, this technology can transform financial services, making them more efficient and customer-focused.

RECOMMENDATIONS

By leveraging the real-time data generated, Bank of Kigali can establish a fraud detection system that analyzes transactions instantly. This system could identify suspicious behaviors, such as unusual spending or high-value withdrawals, and alert the bank for immediate action, minimizing the risk of fraud. Machine learning could further refine the detection system to adapt to new fraud patterns.

Additionally, real-time transaction data could enhance personalized marketing. By analyzing customer spending patterns, the bank can create targeted offers based on their current activities, such as timely loan or credit card offers. Providing real-time account balance updates would also improve user experience, increasing trust and engagement.

Finally, the bank should consider integrating real-time customer feedback through automated surveys post-transaction. This would enable prompt responses to customer concerns, improving both operational efficiency and customer satisfaction through proactive engagement.

REFERENCES

- [1]. Abba, S., & Garba, A. M. . (2019). An IoT-based smart framework for a human heartbeat rate monitoring and control system. Multidisciplinary Digital Publishing Institute Proceedings.
- [2]. david. (2023). *realtime notification*. Retrieved from <https://ultimatemember.com/extensions/real-time-notifications/#:~:text=Add%20a%20real%2Dtime%20notification,when%20user%20role%20is%20changed>.
- [3]. Erin Brenner,Stan Carey. (2023). *Monitoring*. Retrieved from <https://www.vocabulary.com/dictionary/monitoring>
- [4]. Hassanalieragh, M., Page, A., Soyata, T., Sharma, G., Aktas, M., Mateos, G., ... & Andreescu, S. . (2015). Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges. . IEEE international conference on services computing (pp. 285-292). IEEE.

- [5]. Malijan, R. (2013, November 18). *Related Literature and Studies*. Retrieved from <https://www.slideshare.net/RoquiMalijan/group-3-28367418>
- [6]. Nurdin, M. R. F., Hadiyoso, S., & Rizal, A. . (2016). A low-cost Internet of Things (IoT) system for multi-patient ECG's monitoring. September). A low-cost Internet of Things (IoT) system for multi-patient ECG's monitoring. In 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC) .
- [7]. Saddam. (2015, november 4). *Automatic Water Level Indicator and Controller using Arduino*. Retrieved june 29, 2023, from circuitdigest: <https://circuitdigest.com/microcontroller-projects/water-level-indicator-project-using-arduino>
- [8]. wikipedia. (2023). *what is alerting*. Retrieved from https://en.wikipedia.org/wiki/Alerting_system
- [9]. wikipedia. (2024, january 06). *Meaning of Health_care*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Health_care#:~:text=For%20other%20uses,and%20individuals%2C%20influenced