Guru Gobind Singh Indraprastha University
March 2025

Real Time Stock Data Streaming and Ingestion [ELT]

Submitted in partial fulfilment of the requirements

for the award of the degree of

Bachelor of Computer Application

Submitted to:
Dr. Narinder Kaur

Submitted by:
Varun Kumar (06190302022)
Pratham Jindal (09190302022)

Section: BCA M2

ACKNOWLEDGEMENT

We want to express my sincere gratitude to everyone who helped to finish this Data Engineering project. We want to start by expressing my gratitude to Dr. Narinder Kaur for all of her help and support during this educational journey. Her extensive experience and profound understanding have greatly influenced the course and result of this project.

We are grateful to our peers and colleagues who have offered their support and encouragement, contributing to a collaborative and stimulating learning environment. The exchange of ideas and constructive discussions have greatly enriched our project.

Thank you all for your invaluable contributions to this project.

Varun Kumar

Pratham Jindal

CERTIFICATE

This is to certify that this project entitled **Real Time Stock Data Streaming and Ingestion [ELT]**submitted in partial fulfillment of the degree of Bachelor of Computer Applications to the Institute of Innovation in Technology and Management through Department of Computer Application done by Mr. Varun Kumar and Mr. Pratham Jindal Roll No. 06190302022 and 09190302022 is an authentic work carried out by them at under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Varun Kumar

Pratham Jindal

Dr. Narinder Kaur

CERTIFICATE

This is to certify that the dissertation/project report entitled **Real Time Stock Data Streaming and Ingestion [ELT]** is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of Dr. Narinder Kaur. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Signature of the student

Varun Kumar (06190302022)
Pratham Jindal (09190302022)

INDEX

SYNOPSIS

Problem Statement

The stock market generates a vast amount of real-time data, making it challenging to efficiently store, process, and analyze trends. Traditional relational databases struggle with large-scale, high-frequency financial data due to latency and performance bottlenecks.
This project addresses the need for a real-time, scalable, and efficient system to process and visualize stock market data using modern technologies like Kafka, ClickHouse, and Apache Superset.

Why This Topic Was Chosen

Financial markets play a crucial role in the global economy, and data-driven insights are essential for investors, analysts, and traders. Traditional methods of stock analysis are often slow and inefficient, requiring modern solutions that can handle large volumes of data in real-time.
By implementing Kafka for stream processing, ClickHouse for optimized storage, and Superset for visualization, this project aims to provide a high-performance platform for stock market analysis.

Objectives

- Real-time Data Collection: Fetch live stock market data from Yahoo Finance.
- Stream Processing with Kafka: Use Kafka as a distributed messaging system to ensure data integrity and scalability.
- Efficient Data Storage with ClickHouse: Leverage ClickHouse for high-speed querying and storage of stock data.
- Data Visualization with Superset: Provide an interactive dashboard for stock market insights.
- Scalability & Performance: Ensure the system handles high-frequency trading data efficiently.

Introduction

The financial market generates an enormous amount of stock data daily, requiring efficient methods for real-time storage, processing, and analysis.
This project aims to build a Stock Data Analysis System that fetches live stock data using Yahoo Finance (yfinance), processes it via Apache Kafka, and stores it efficiently in ClickHouse, a high-performance columnar database.
The system will be visualized using Apache Superset, enabling traders, investors, and analysts to make data-driven decisions.

Scope of the Project

This project will benefit financial analysts, traders, and investors by providing:

- A real-time view of stock market trends.
- Fast data retrieval for historical analysis.
- An interactive dashboard for visualizing stock performance.

Methodology

1. **Data Collection**
    - Utilize Yahoo Finance API (yfinance) to fetch stock data, including open, close, high, low prices, and trading volume.
    - Set up a Kafka producer to stream this data in real time.
2. **Data Processing**
    - Use Kafka consumer to consume stock data.
    - Store the processed data in ClickHouse, optimizing for fast retrieval and aggregation.
3. **Data Visualization**
    - Integrate Apache Superset to create an interactive dashboard.
    - Display key stock market trends, historical price movements, and volume analysis.

4. **Technology Stack**
    - Backend: Python, Kafka (Producer & Consumer)
    - Database: ClickHouse
    - Data Source: Yahoo Finance API
    - Visualization: Apache Superset
    - Deployment: Docker (for ClickHouse, Kafka, and Superset)

Expected Outcome

- A real-time stock analysis system with efficient data ingestion and querying.
- A scalable, high-performance financial data pipeline.
- An interactive visualization dashboard for stock market trends.

Future Enhancements

- Implement predictive analytics for stock price forecasting using machine learning.
- Add user authentication for personalized watchlists and alerts.
- Expand data sources to include cryptocurrency and forex market data.

Conclusion

The Stock Data Analysis System will provide an efficient, real-time, and scalable approach to handling financial market data.
The combination of Kafka, ClickHouse, and Superset ensures high-speed data processing, storage, and visualization, making it a valuable tool for financial professionals and investors.

Chapter 1: Objective and Introduction

1.1Problem Statement

Stock market investors and analysts require real-time stock price data to make informed trading decisions. Traditional systems often suffer from latency issues, inefficient data storage, and limited analytical capabilities. There is a need for a fast, scalable, and interactive solution that provides real-time stock price updates along with historical analysis for better decision-making.

1.2 Why This Topic Was Chosen:

This topic was chosen because real-time financial data analysis is crucial for investors, traders, and analysts. Stock markets are highly dynamic, and having fast, accurate, and scalable data streaming solutions can provide a competitive edge. With the increasing importance of data-driven decision-making, this project leverages Kafka, ClickHouse, and Power BI to create a robust system for real-time stock price monitoring and analysis.

1.3 Introduction:

The stock market is a dynamic environment where prices fluctuate rapidly based on **market trends, company performance, and economic factors**. Investors, traders, and analysts rely on real-time stock data to make **informed financial decisions**.

This project aims to develop a **real-time stock price monitoring and analysis system** using **Kafka, ClickHouse, and Power BI**.

The system will:

Stream live stock prices using Kafka and store them in ClickHouse.

Provide historical stock trend analysis to identify patterns.

Visualize real-time and past data using Power BI dashboards.

Unlike traditional stock market applications, this system is **optimized for high-speed data processing, real-time analytics, and scalability**.

1.4Scope of the Project:

1. **Real-Time Data Streaming**
   - Stock price data will be fetched using **Yahoo Finance (yfinance)**.
   - Kafka will be used for **real-time streaming** to ensure low-latency data flow.
2. **Efficient Data Storage and Processing**
   - ClickHouse will store stock data for **fast querying and retrieval**.
   - Data will be structured for **efficient analysis of real-time and historical trends**.
3. **Interactive Data Visualization**
   - Power BI dashboards will display:
     - **Live stock prices** (current price, open, close, high, low).
     - **Historical trends** (price fluctuations over time).
     - **Market analysis** (sector-wise performance, volume trends).
4. **Scalability & Future Enhancements**
   - Integration of **machine learning models** for stock price predictions.
   - API development for **third-party applications** to access real-time stock data.

1.5 Methodology

1.5.1 Data Collection

Stock market data is highly dynamic and requires efficient real-time collection and processing. In this project, stock prices are fetched using the yfinance Python library, which provides access to historical and real-time stock data from Yahoo Finance.

- The collected data includes:
- Ticker Symbol (Stock Identifier)
- Open Price, Close Price, High, Low
- Trading Volume
- Timestamp (Date & Time of Data Retrieval)

To ensure real-time streaming, an Apache Kafka producer fetches the stock prices at regular intervals and sends them to a Kafka topic. A Kafka consumer then reads this data and inserts it into ClickHouse, a high-performance columnar database optimized for analytical workloads.

1.5.2 SDLC Model

The Software Development Life Cycle (SDLC) followed in this project is the Agile Model, which emphasizes:

- Iterative development with continuous feedback
- Flexibility to accommodate changes in requirements
- Frequent testing and integration to ensure stability

The key phases include:

- Requirement Analysis – Identifying the need for a real-time stock price visualization system.
- System Design – Defining Kafka, ClickHouse, Power BI, and Docker components.
- Implementation – Developing Kafka producer/consumer, setting up ClickHouse, and integrating Power BI.
- Testing – Ensuring correct data ingestion, storage, and visualization.
- Deployment – Running the project in a Dockerized environment.
- Maintenance & Improvements – Iterative refinements based on performance and user feedback.

Category	Technology Used
Programming Language	Python
Data Streaming	Apache Kafka
Database	ClickHouse/Apache Druid
Visualization	Power BI/SuperSet
Libraries	Pandas, yfinance
Deployment	Docker

Table 1.1: Technologies and Tools during planning.

1.5.3 Gantt Chart

A Gantt Chart is used to track project timelines and task dependencies. Below is a sample breakdown of the project timeline:

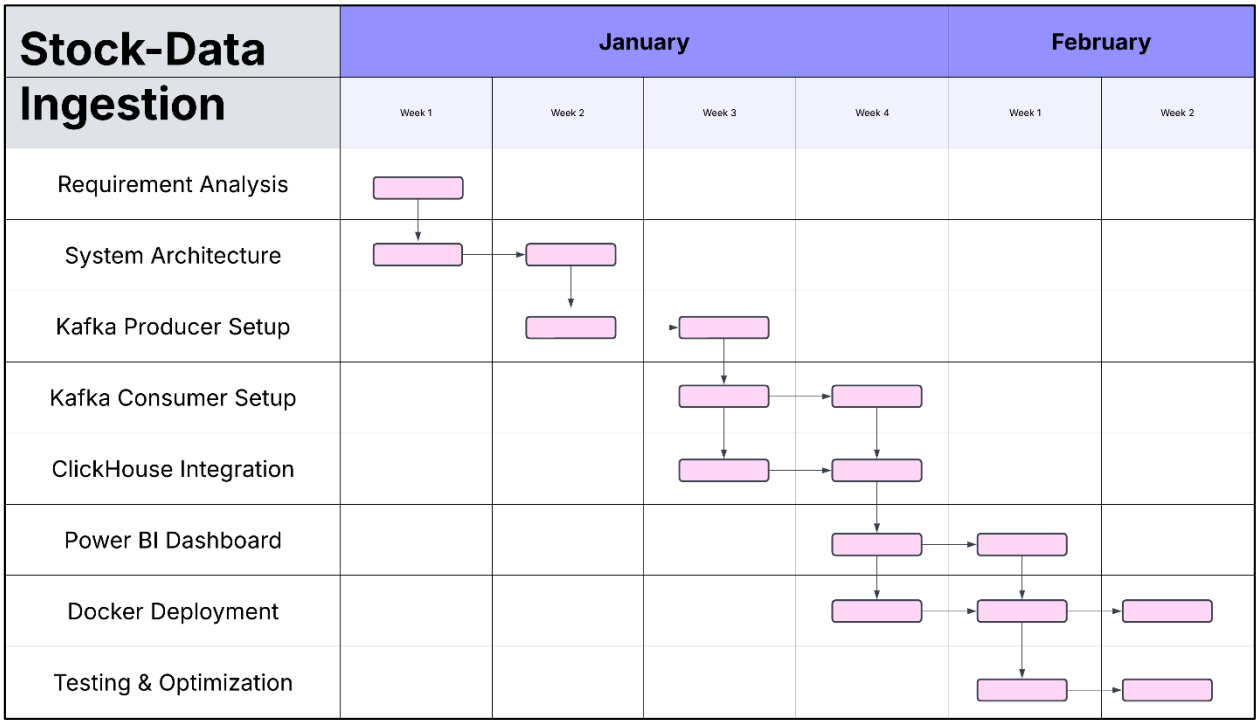

Fig 1.1: Gantt Chart

1.6 Conclusion:

This project demonstrates a **robust and scalable approach** to **real-time stock price visualization**. By combining **Kafka, ClickHouse, and Power BI**, the system enables **high-speed data streaming, efficient storage, and interactive visualization**.

With potential improvements such as **AI-driven predictions and automated alerts**, this system can further enhance stock market analysis and decision-making for investors and analysts.

Chapter 2: Software Requirements Specification (SRS)

The Stock Market and the Need for Real-Time Data

The stock market is a dynamic and fast-paced financial environment where prices fluctuate constantly due to market trends, economic indicators, company performance, and global events. Investors, traders, and financial analysts rely on real-time data to make informed trading decisions. The ability to access, process, and visualize live stock prices is crucial in minimizing risks and maximizing profits.

Traditional stock price monitoring systems often face challenges such as latency, inefficient storage, and limited analytics capabilities. This project aims to address these challenges by using Apache Kafka for real-time data streaming, ClickHouse for high-speed storage and analytics, and Power BI for interactive visualization.

2.1 Introduction

This document outlines the Software Requirements Specification (SRS) for the stock price streaming and visualization project. It defines the project's functionality, system architecture, user requirements, and technical specifications to ensure a well-structured and efficient implementation.

2.2 Purpose

The purpose of this system is to provide a real-time stock price streaming and visualization solution using Kafka, ClickHouse, and Power BI. The system aims to:

- Continuously collect stock price data from yfinance.
- Stream the data through Apache Kafka.
- Store and process data efficiently in ClickHouse.
- Visualize stock trends using Power BI for real-time decision-making.

2.3 Scope

This system is designed for:

- Stock traders and analysts who require real-time insights.
- Financial institutions looking to analyze stock trends.
- Developers and researchers interested in streaming architectures.

The system will:

- Support multiple stock symbols from Yahoo Finance.
- Stream data at configurable intervals using Kafka.
- Store structured stock data in ClickHouse.
- Provide real-time and historical visualizations in Power BI.

2.4 Key Challenges in Stock Price Monitoring and Analysis

2.4.1. Latency in Real-Time Data Processing

Stock prices change within **fractions of a second**, making low-latency data processing crucial. Traditional database systems may struggle to keep up with **millions of transactions per second**, leading to delayed insights and missed trading opportunities.

2.4.2. Efficient Storage and Retrieval of Large Datasets

Stock market data is **highly voluminous**, consisting of **price movements, trading volumes, bid-ask spreads, and historical trends**. Efficient storage solutions must **support fast queries** to provide actionable insights without performance bottlenecks.

2.4.3. Real-Time and Historical Trend Analysis

Financial decision-making depends on both **real-time price changes** and **historical trends**. Traders need tools to:

- Compare **current prices with historical averages**.
- Identify **patterns and market trends** over time.
- Analyze **sector-wise performance** for diversified investments.

2.4.4. Data Visualization for Decision-Making

Raw stock price data can be difficult to interpret. **Power BI dashboards** can help by displaying:

- **Live price movements** with interactive charts.
- **Historical performance** trends.
- **Sector-wise market insights** for better decision-making.

2.4.5. Scalability and Performance Optimization

A stock price monitoring system must handle **high-frequency trading data** and scale efficiently as market activity increases. The architecture must support **real-time ingestion, fast queries, and scalable storage**.

2.5 How the System Works in Practice

2.5.1. Data Collection (Kafka Producer)

- Stock price data is fetched from **Yahoo Finance (yfinance)**.
- The **Kafka producer** streams real-time stock data to a Kafka topic.

2.5.2. Data Storage (Kafka Consumer & ClickHouse)

- The **Kafka consumer** retrieves streamed data and **stores it in ClickHouse**.
- ClickHouse is optimized for **fast analytics and real-time queries**.

2.5.3. Data Processing & Analysis

- The system processes **price fluctuations, trading volumes, and market trends**.
- Historical data is **aggregated** to provide long-term insights.

2.5.4. Visualization & Decision Support (Power BI Dashboard)

- Power BI provides **real-time stock price visualization**.
- Users can view **interactive charts and sector-wise trends**.
- Historical price comparisons help investors make **data-driven decisions**.

2.6 Functional Requirements

2.6.1 Data Collection & Streaming

- The system shall fetch stock price data using yfinance.
- The Kafka producer shall publish stock price data to a Kafka topic.
- The Kafka consumer shall read stock data and store it in ClickHouse.

2.6.2 Storage & Querying

- The system shall store stock price data in ClickHouse.
- The database shall support fast analytical queries for historical trends.
- The system shall handle large volumes of stock data efficiently.

2.6.3 Visualization & Reporting

- The system shall integrate ClickHouse with Power BI for dashboarding.
- The Power BI reports shall support real-time stock monitoring.
- Users shall be able to filter and analyze stock trends interactively.

2.7 Non-Functional Requirements

2.7.1 Performance

- The system shall process stock price data with low latency.
- ClickHouse queries shall return results within seconds for large datasets.

2.7.2 Scalability

- The system shall support multiple stock tickers without performance degradation.
- Kafka shall handle high-throughput streaming efficiently.

2.7.3 Reliability & Availability

- The system shall be highly available and fault-tolerant.
- If Kafka fails, the system shall retry failed messages automatically.

2.7.4 Security

- Data access shall be restricted via user authentication in Power BI.

- Database connections shall use secure authentication mechanisms.

2.8 System Architecture Overview

2.8.1 Dockerized Environment

Since some tools are not directly compatible with Windows, we containerized our services using Docker Compose. The following services have been defined:

- Zookeeper: Manages Kafka brokers.
- Kafka: Handles real-time stock price data streaming.
- ClickHouse: Stores processed stock data efficiently.
- Superset (Optional): Alternative data visualization tool.

2.8.2 Kafka Producer

A Python-based Kafka Producer fetches stock prices from Yahoo Finance (yfinance) and streams them to Kafka topics. The producer ensures:

- Real-time stock price updates for selected tickers.
- Inclusion of metadata such as company sector, industry, country, and past historical prices.
- Fault-tolerant design with retry mechanisms.

2.8.3 Kafka Consumer

A Kafka Consumer listens to stock price updates and inserts data into ClickHouse. It:

- Parses real-time messages from Kafka.
- Inserts data into two ClickHouse tables:
    - live_stock_data (for current market prices).
    - historical_stock_data (for monthly reference prices).
- Handles message consistency and error recovery.

Chapter 3: System Design and Analysis

3.1 Analysis Methodology

3.1.1 Exploratory Data Analysis (EDA):

Before implementing the system, an in-depth analysis of stock price data is essential to identify trends, patterns, and anomalies.

Understanding the Data:

- Stock price datasets include open price, close price, high, low, volume, bid price, ask price, and timestamps.
- Visualizing trends using histograms, line charts, and scatter plots helps identify price fluctuations and volatility.

Identifying Missing Values:

- Missing data in stock prices (e.g., holidays or market downtime) is handled by forward-fill, interpolation, or removal.

Detecting Outliers:

- Sudden price spikes or anomalies can indicate market manipulation, earnings reports, or macroeconomic events.
- Outliers are detected using box plots, Z-score analysis, and moving averages.

3.2 Hardware Requirements

3.2.1 Processor: Multi-core CPUs (Intel i7, AMD Ryzen 7) to handle **real-time data streaming** and ClickHouse queries efficiently.

3.2.2 RAM: At least **16GB of RAM** to support **high-speed data ingestion and real-time analytics**.

\

3.2.3 Storage:

- **NVMe SSD (512GB or higher)** for **fast read/write operations** in ClickHouse.
- **Additional storage** for historical stock price data.

3.2.4 Network: High-speed internet for **low-latency data retrieval from Yahoo Finance (yfinance)**.

3.3 Software Requirements

Programming Language:

- Python for data collection, streaming, and analytics.

Data Processing & Streaming:

- Apache Kafka: Real-time stock price streaming.
- ClickHouse: High-speed database for stock price storage.

Data Visualization:

- Power BI: Interactive stock price dashboards.

Deployment & Environment:

- Docker: Containerized deployment for Kafka, ClickHouse, and Power BI Gateway.

3.4 Data Analysis & Processing Libraries

- **Pandas & NumPy:** Data manipulation, time-series analysis.
- **yFinance API**: Fetches real-time stock price data.
- **Kafka-Python**: Handles real-time data streaming.
- **SQLAlchemy:** Connects Python with ClickHouse for efficient queries.

3.5 Block Diagram

A block diagram provides a high-level overview of the system architecture. It illustrates how different components interact within the stock price streaming and visualization pipeline.

System Components:

- Stock Data Source → Yahoo Finance (yfinance)
- Kafka Producer → Fetches stock prices and publishes them to Kafka
- Kafka Broker → Manages message queues
- Kafka Consumer → Reads messages from Kafka and inserts them into ClickHouse
- ClickHouse Database → Stores stock price data for analysis
- Power BI → Connects to ClickHouse for real-time visualization
- Docker → Manages containerized deployment.

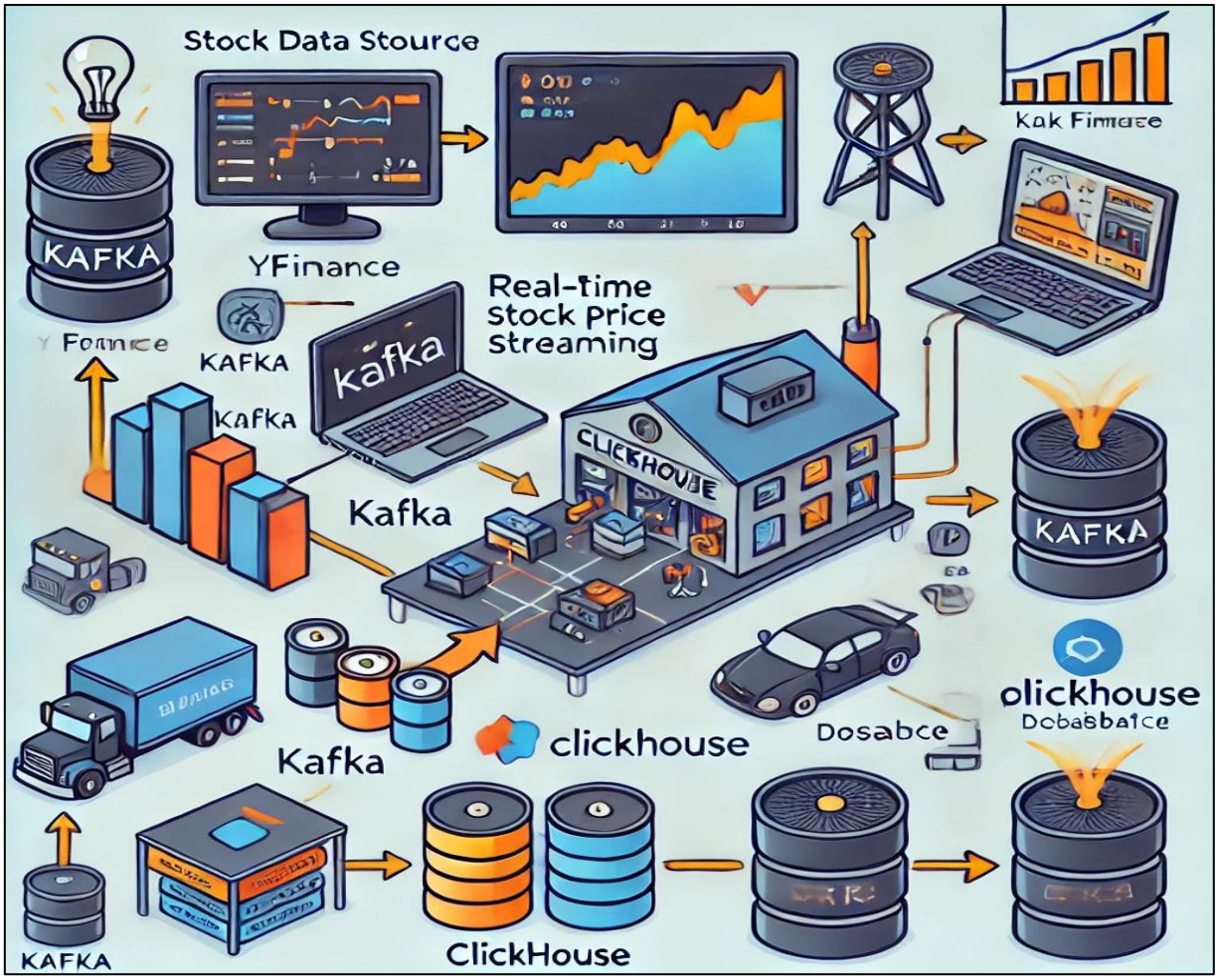

Fig 3.1: Animated Block Diagram

3.6 Database Design

1. Table Schema: live_stock_data

This table is designed to store real-time stock market data, including price movements, company details, and timestamps.

Column Name	Data Type	Description
Name	String	Ticker symbol (e.g., ICICI,HDFC)
Current_Price	Float64	Latest traded stock price
Open_Price	Float64	Opening price of the trading day
Previous_Close	Float64	Price at which the stock closed the previous day
Day_High	Float64	Highest price during the trading session
Day_Low	Float64	Lowest price during the trading session
Bid_Price	Float64	Price buyers are willing to pay
Ask_Price	Float64	Price sellers are willing to accept
Company_Name	String	Full name of the company
Sector	String	Market sector (e.g., Technology, Finance)
Industry	String	Specific industry within the sector
Country	String	Country where the company is based
Timestamp	UInt64	Unix timestamp of data ingestion
Formatted_Timestamp	DateTime('Asia/Kolkata')	Human-readable timestamp
Volume	Nullable(UInt64)	Number of shares traded (nullable)

Table 3.1: Live Stock Database Design

Key Features:

- **Real-time stock data** is inserted continuously via **Kafka consumers**.
- **Formatted_Timestamp** is stored in the **Asia/Kolkata timezone** for consistency.
- **MergeTree Engine** enables **fast read/write operations** and **efficient querying**.

3.7 Data Flow Diagram (DFD) - Level 0

This diagram represents a high-level view of the system, showing how data moves between key components:

- User requests stock price data via Power BI.
- Kafka Producer fetches stock data from yfinance.
- Kafka Broker manages data flow between producer and consumer.
- Kafka Consumer inserts data into ClickHouse.
- Power BI fetches data from ClickHouse and displays it in dashboards.

3.7.1 Data Flow Process

- The **Kafka Producer** fetches live stock data and pushes it into respective Kafka topics.
- The **Kafka Consumer** listens to these topics and stores the data in ClickHouse.
- Power BI connects to ClickHouse via an ODBC driver to fetch structured data.
- **Power BI Dashboards** visualize:
  - **Total Volume Traded & Current Price Trend**: A combination of bar and line charts showing volume and price movement over time.
  - **Candlestick Chart**: A technical analysis chart displaying open, high, low, and close (OHLC) prices.

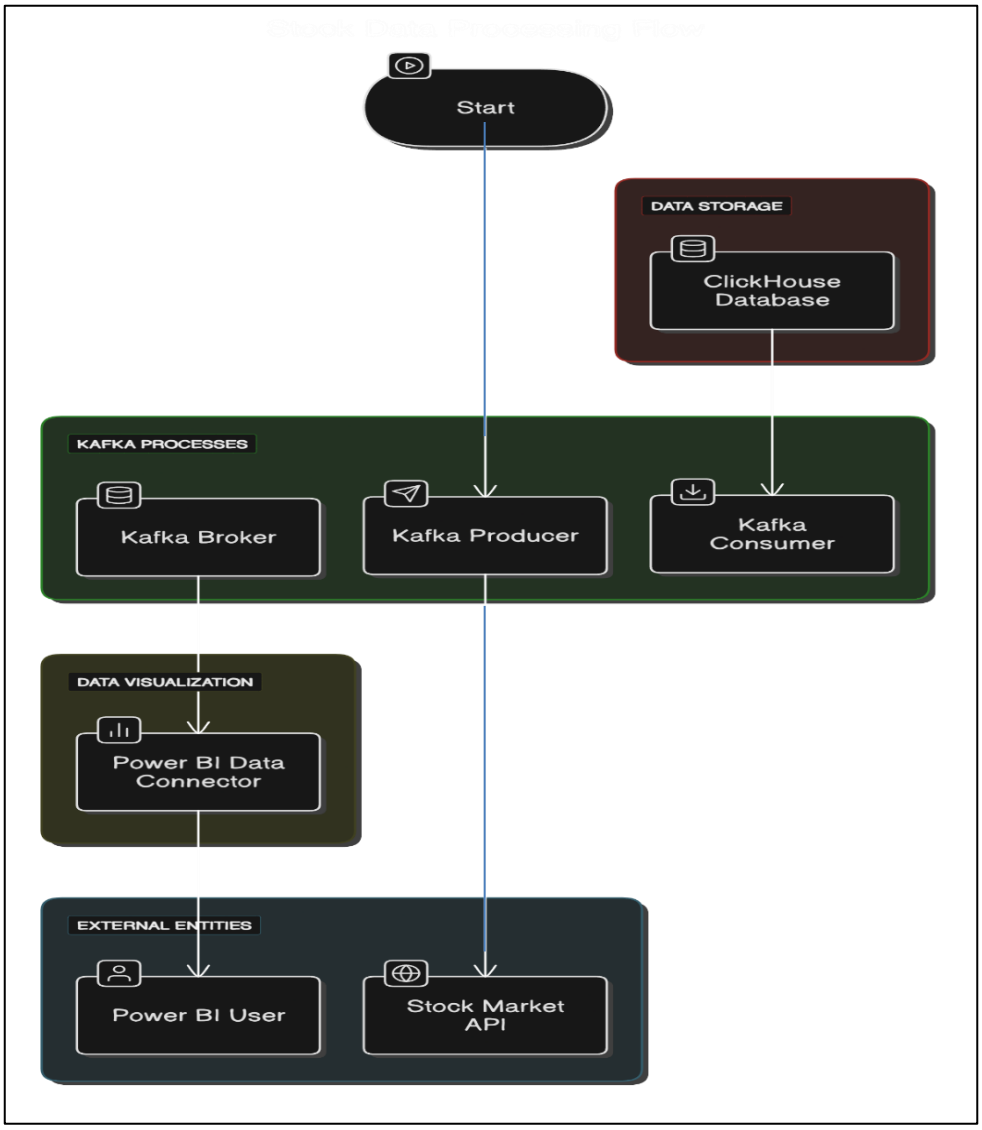

Fig 3.2: User Flow Diagram

3.8 Site Map

As the project does not involve a web interface, a traditional site map is not applicable. Instead, the Power BI dashboard layout is structured as follows:

- Real-time Stock Prices – Displays current price, open, close, high, and low values.
- Historical Trends – Tracks price fluctuations over time.
- Stock Market Analysis – Provides sector-wise performance and volume trends.

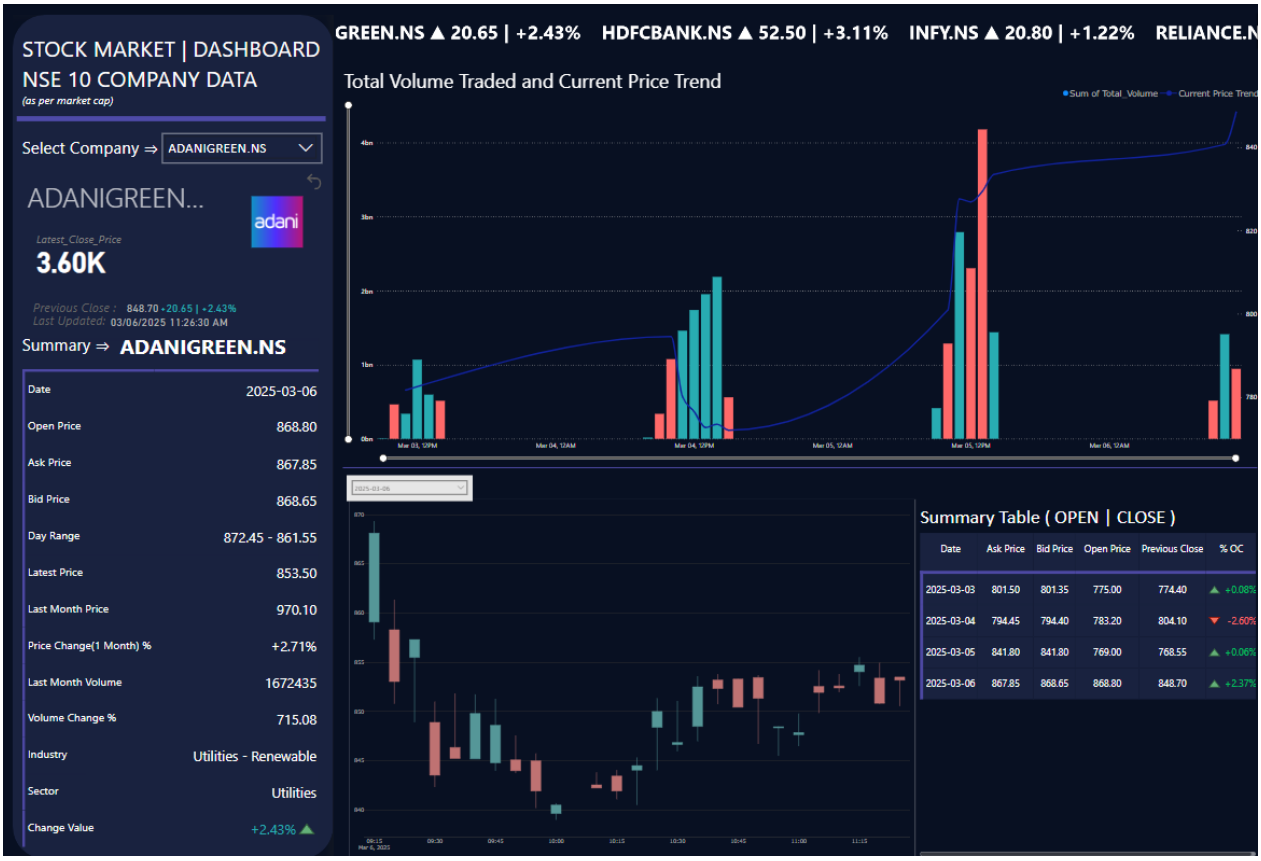

Fig 3.3: PowerBi Dashboard

Chapter 4: Software Development

4.1 Designing of Front Pages

Since this project focuses on data streaming and visualization, the front-end interface is built using Power BI, rather than a conventional web-based UI. The Power BI dashboard provides an interactive and analytical view of stock data in real time.

Power BI Dashboard Layout

The Power BI dashboard is structured into the following sections:

1. Real-time Stock Prices
    - Displays current stock prices with key metrics:
        - Current Price
        - Open Price
        - Close Price
        - Day High & Low
        - Volume
        - Live updates ensure the latest market trends are reflected instantly.
2. Historical Trends
    - Line and Candlestick Charts to visualize stock price movements over time.
    - Users can apply filters for custom timeframes (e.g., daily, weekly, monthly).
3. Stock Market Analysis
    - Sector-wise performance comparison to identify market trends.
    - Volume analysis to track trading activity.
4. Filters and Customization
    - Users can select specific stocks, sectors, or timeframes.
    - Date range filters allow analysis of different periods.

The dashboard is designed for high usability, ensuring traders and analysts can quickly interpret market trends.

4.2 Database Design

The system uses ClickHouse, a high-performance analytical database, to store and process large volumes of stock price data.

ClickHouse Table: live_stock_data

The main table,[live_stock_data], is structured as follows:

Column Name	Data Type	Description
Name	String	Stock ticker symbol (e.g., AAPL, TSLA)
Current_Price	Float64	Latest traded stock price
Open_Price	Float64	Opening price of the trading day
Previous_Close	Float64	Previous day's closing price
Day_High	Float64	Highest price during the trading session
Day_Low	Float64	Lowest price during the trading session
Bid_Price	Float64	Price buyers are willing to pay
Ask_Price	Float64	Price sellers are willing to accept
Company_Name	String	Full name of the company
Sector	String	Market sector (e.g., Technology, Finance)
Industry	String	Specific industry classification
Country	String	Country where the company is based
Timestamp	UInt64	Unix timestamp of data ingestion
Formatted_Timestamp	DateTime('Asia/Kolkata')	Readable timestamp
Volume	Nullable(UInt64)	Number of shares traded (nullable)

Table 4.1: Database Design Description

Optimizations for Fast Query Execution

To ensure efficient data retrieval and analysis, the following optimizations are applied:

- Storage Engine: MergeTree() – Optimized for high-performance storage and analytical queries.
- Ordering Strategy: ORDER BY (Name, Timestamp) – Enables efficient filtering by stock symbol and date.
- Partitioning: PARTITION BY toYYYY-MM(Formatted_Timestamp) – Organizes data into monthly partitions, improving query performance.

4.3 Connectivity

The system integrates multiple components to enable real-time stock price streaming and visualization.

Data Flow Between System Components

1. Stock Data Collection (yfinance)
   - A Kafka Producer fetches real-time stock prices from yfinance.
   - Data is published to a Kafka Topic for further processing.
2. Kafka for Streaming
   - Kafka Broker manages message queues and ensures reliable delivery.
   - Kafka Consumer subscribes to the topic and reads stock price updates.
3. Data Storage in ClickHouse
   - The Kafka Consumer inserts stock data into the live_stock_data table in ClickHouse.
   - This ensures a continuous flow of incoming stock price updates.
4. Power BI Integration
   - Power BI connects to ClickHouse using an ODBC driver.
   - Users can query and visualize real-time stock data in Power BI dashboards.

Technology Stack

Component	Technology Used
Data Source	yfinance (Yahoo Finance API)
Streaming	Apache Kafka
Database	ClickHouse
Visualization	Power BI
Deployment	Docker

Table 4.2: Technological Stack

4.4 System Planning

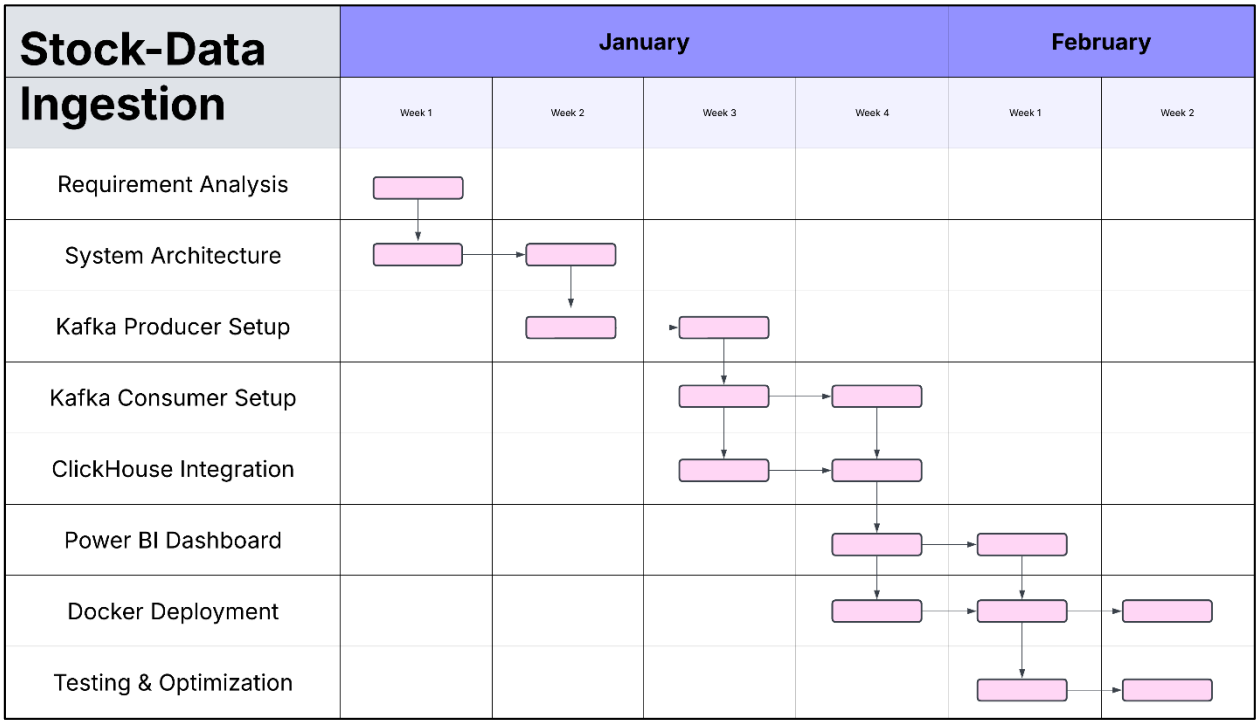

Fig 4.1: Gantt Chart

The development of the Stock Data Ingestion System follows a structured and phased approach to ensure a comprehensive, functional, and scalable solution. Below is an in-depth explanation of each phase in the project lifecycle:

4.4.1 Start (Date: 1-09-24)

The project begins with the initiation phase, where the primary focus is on aligning all stakeholders (project managers, developers, data engineers, and business stakeholders) on the project's vision, objectives, and deliverables. This phase ensures that everyone involved has a clear understanding of the roadmap, project goals, and key requirements. At the end of this phase, a high-level project plan and timeline are finalized, and any preliminary research or scoping work is done to lay the foundation for subsequent phases.

4.4.2 Requirement Gathering

In the requirement gathering phase, the development team works closely with the stakeholders to capture both functional and non-functional requirements. This phase involves detailed discussions on the features, performance expectations, and specific needs of the stock data

ingestion system. Functional requirements focus on core functionalities such as real-time stock price ingestion, data storage, analytics, and visualization. Non-functional requirements address performance, scalability, security, and system resilience.

4.4.3 System Architecture

In this phase, the overall architecture of the stock data ingestion pipeline is designed. The system is structured to handle real-time stock data using a message-driven architecture. The key components include:

- **Kafka Producer** to fetch stock prices using `yfinance`.
- **Kafka Consumer** to store the data into ClickHouse.
- **ClickHouse Database** for efficient storage and querying of stock data.
- **Power BI Dashboard** for visualizing stock trends and analytics.
- **Docker** to containerize and deploy the entire setup for portability and scalability.

4.4.4 Kafka Producer Setup

This phase involves setting up a Kafka producer to fetch real-time stock prices from external sources (e.g., Yahoo Finance API). The producer is responsible for streaming stock data into Kafka topics, ensuring reliability and scalability.

4.4.5 Kafka Consumer Setup

The Kafka consumer is implemented to process messages from Kafka topics and insert them into the ClickHouse database. This ensures efficient ingestion and storage of stock data, enabling high-speed queries and analytics.

4.4.6 Database Design

A well-structured database schema is designed to store stock price data effectively. The key aspects include:

- **Tables for storing raw stock data** (e.g., `live_stock_data` and `candlestick_data`).
- **Indexing strategies** to optimize query performance in ClickHouse.
- **Partitioning and sharding techniques** to handle large-scale data efficiently.

- **Historical data retention policies** to manage storage and performance.

4.4.7 ClickHouse Integration

Integration with ClickHouse ensures high-performance querying and analytics. This phase involves setting up ClickHouse tables, optimizing data ingestion pipelines, and verifying query performance for real-time analytics.

4.4.8 Power BI Dashboard Development

The Power BI dashboard is developed to visualize real-time and historical stock data. The key features include:

- Live stock price tracking.
- Candlestick chart visualization.
- Custom analytics and trend detection.
- User-friendly filters and interactive reports.

4.4.9 Docker Deployment

The entire system is containerized using Docker to streamline deployment and scalability. This phase includes:

- **Creating Docker containers** for Kafka, ClickHouse, and other components.
- **Orchestrating containers** for smooth integration.
- **Ensuring reproducibility and scalability** in different environments.

4.4.10 Testing and Optimization

The final phase focuses on validating the system's performance, reliability, and usability:

- **Unit Testing:** Ensuring each component (Kafka producer, consumer, ClickHouse queries) functions correctly.
- **Integration Testing:** Verifying that all system components work together as expected.
- **Performance Testing:** Assessing system scalability under high data loads.
- **User Acceptance Testing (UAT):** Ensuring the Power BI dashboard meets stakeholder expectations.

4.4.11 Final Deployment

Once testing is complete, the system is deployed in a production environment. Scheduled refreshes and gateway configurations for Power BI are set up to enable real-time data updates. Continuous monitoring and optimization are performed to ensure efficient stock data ingestion and visualization.

This structured development process ensures that the Stock Data Ingestion System meets the business requirements while maintaining efficiency, scalability, and reliability.

4.5 System User Interface

The Stock Data Summary and Analysis chapter provides an overview of the key stock market metrics and trends observed in the analyzed dataset. This section includes a structured representation of stock price movements, trading volumes, and key indicators, allowing for an in-depth understanding of market dynamics. The data is sourced from real-time stock feeds and processed through the stock ingestion system to ensure accuracy and relevance.

4.5.1 Stock Data Overview

The stock dashboard presents vital financial statistics, including:

- **Open Price**: The initial trading price of the stock at market opening.
- **Ask Price**: The price at which sellers are willing to sell the stock.
- **Bid Price**: The price at which buyers are willing to purchase the stock.
- **Day Range**: The highest and lowest price fluctuations within a given trading day.
- **Latest Price**: The most recent recorded trading price.
- **Previous Close**: The last recorded price from the previous trading session.
- **Volume Traded**: The total number of shares exchanged during a trading period.
- **Percentage Change**: The rate of price fluctuation compared to previous closing values.

These metrics provide essential insights into stock performance, helping traders and investors make informed decisions.

4.5.2 Summary Table Analysis (Open vs Close Prices)

The summary table presents key stock price movements for selected trading days, highlighting the correlation between open and close prices. The table includes:

Summary Table (OPEN | CLOSE)

Date	Ask Price	Bid Price	Open Price	Previous Close	% OC
2025-03-03	1,742.00	1,741.90	1,739.80	1,732.40	▲ +0.43%
2025-03-04	1,712.55	1,712.00	1,739.80	1,701.55	▲ +2.25%
2025-03-05	1,710.15	1,709.85	1,701.95	1,710.00	▼ -0.47%

Fig 4.2: Summary Table of Open and Close Prices

- A positive **% OC Change** (green) signifies a closing price higher than the previous close, indicating a bullish trend.
- A negative **% OC Change** (red) indicates a drop in stock price from the previous close, signaling a bearish trend.

4.5.3 Visual Trends in Stock Performance

The dashboard includes interactive visualizations that depict key market movements:

Total Volume Traded and Current Price Trend

A combination of bar and line charts represents:

- The **total trading volume** for each time interval.
- The **price trend** throughout the selected time frame.
- A visible correlation between spikes in trading activity and fluctuations in stock prices.

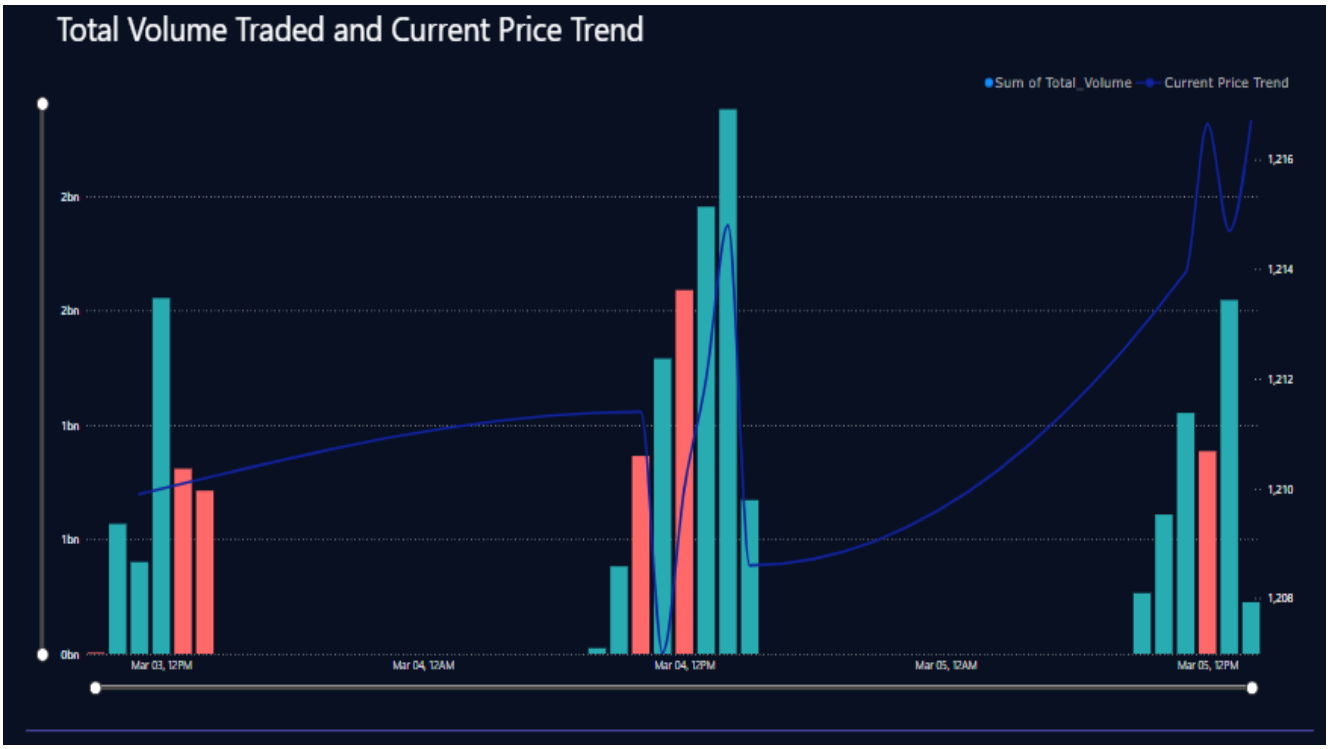

Fig 4.3: Total Volume and Current Price Trend

The chart presents a dual-axis visualization of stock trading activity over time. It consists of:

- **Bar Chart (Left Y-Axis, Cyan & Red Bars):** Represents the total volume of stocks traded at different time intervals.
- **Line Graph (Right Y-Axis, Blue Line):** Displays the trend of the current stock price over the same period.

The bars indicate trading volume, with higher bars signifying increased trading activity. The blue line represents the price trend, showing fluctuations in stock value corresponding to volume changes. The combination of both elements provides insights into the relationship between trade volume and stock price movement.

Candlestick Chart Analysis

Fig 4.4: CandleStick Chart for stocks on 5 min format

The candlestick chart provides an intuitive representation of price movements, with:

- **Wicks (Shadows)** indicating high and low price extremes.
- **Body** representing the difference between open and close prices.
- **Color Coding** (green for bullish trends, red for bearish trends) highlighting price movement patterns.

The chart shown is a candlestick chart, widely used in stock market analysis to represent price movements over time. Each candlestick provides four key data points:

- **Open Price:** The price at which the stock started trading in the given time interval.
- **Close Price:** The price at which the stock ended trading in that interval.
- **High Price:** The highest traded price during the interval.
- **Low Price:** The lowest traded price during the interval.
- **Green (Bullish) Candlesticks:** Indicate that the stock closed higher than it opened.
- **Red (Bearish) Candlesticks:** Indicate that the stock closed lower than it opened.

The vertical lines (wicks) extending from the body show the high and low prices within each interval. This chart helps traders understand market trends, price reversals, and volatility.

Chapter 5: System Testing

System testing ensures that all components of the stock price streaming and visualization system function correctly, meet requirements, and provide a seamless user experience. The testing process involves verifying system functionality, database performance, and the Power BI dashboard's usability.

5.1 Functional Testing

Tested Components:

1. Kafka Producer & Consumer
    - Stock price data is fetched from yfinance without errors.
    - Messages are successfully published to Kafka topics.
    - Kafka Consumer reads and inserts data into ClickHouse correctly.
2. ClickHouse Database
    - Data is stored accurately with correct timestamps.
    - Queries return the expected stock price data without delay.
3. Power BI Dashboard
    - Power BI successfully connects to ClickHouse and retrieves data.
    - Dashboard filters (e.g., stock symbol, date range) work correctly.
    - Charts and reports update as expected with new data.

5.2 Database Testing

Database testing ensures data integrity, query performance, and correctness in ClickHouse.

Test Cases:

- Data Insertion Test → Kafka Consumer successfully inserts stock data into ClickHouse.
- Data Retrieval Test → Queries return accurate stock prices.
- Performance Test → Query response times remain under expected thresholds.
- Data Consistency Check → No missing or duplicate records.

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

The stock ingestion system effectively addresses the challenge of efficiently streaming, storing, and processing real-time stock market data. By leveraging a robust architecture involving Kafka for data streaming, ClickHouse for high-performance storage, and a consumer service to ingest and structure stock price data, the system ensures scalability, reliability, and low-latency data processing.

The project's backend integrates real-time data fetching using APIs, ensuring continuous updates for stock prices. The structured pipeline allows for seamless integration with visualization tools like Power BI and Grafana, providing actionable insights for traders, analysts, and investors.

6.2 Key Achievements Include:

6.2.1 Real-Time Data Ingestion:

The system continuously fetches stock price data using a Kafka producer, enabling a steady stream of updates for accurate analysis.

6.2.2 High-Performance Storage:

ClickHouse enables efficient storage and querying of time-series data, optimizing for fast retrieval and historical analysis.

6.2.3 Scalable and Modular Architecture:

The modular design ensures easy integration with additional data sources, APIs, and analytics platforms as needed.

Overall, this project demonstrates the practical application of event-driven architectures and modern database technologies to handle high-frequency stock market data ingestion and visualization effectively.

6.3 Future Scope

Several enhancements can further improve the efficiency, accuracy, and usability of the stock ingestion system:

6.3.1 Advanced Data Processing and Analytics

- Machine Learning for Trend Prediction: Implement predictive analytics using machine learning models to detect trends, anomalies, and potential investment opportunities.
- Sentiment Analysis Integration: Incorporate sentiment analysis from news or social media sources to provide additional insights into stock movements.

6.3.2 Real-Time Enhancements

- Low-Latency Data Processing: Optimize the Kafka consumer for even lower latency to support real-time trading strategies.
- Event-Driven Alerts: Develop automated alerts for price movements, volume spikes, or other key market indicators.

6.3.3 Data Enrichment

- Additional Market Data Sources: Integrate data from multiple exchanges and financial news sources to enhance decision-making.
- Fundamental Data Integration: Combine financial reports, earnings announcements, and macroeconomic indicators for a comprehensive stock analysis.

6.3.4 Visualization and User Experience

- Interactive Dashboards: Enhance Grafana and Power BI dashboards with more customizable visualizations and user-friendly interactions.
- Mobile Compatibility: Develop a mobile-friendly interface or a dedicated app for tracking stock movements on the go.

6.3.5 Scalability and Cloud Integration

- Cloud-Based Deployment: Move the infrastructure to cloud services for enhanced scalability, reliability, and access to managed Kafka and ClickHouse solutions.

- API Access for External Users: Provide API endpoints to allow third-party applications or analysts to access processed stock data.

6.3.6 Security and Compliance

- Data Encryption and Access Control: Implement enhanced security measures to protect sensitive financial data.
- Regulatory Compliance: Ensure adherence to financial regulations and data protection laws for global market compatibility.

Conclusion

The future scope for this stock ingestion system is extensive, with opportunities to enhance functionality, analytics, and user experience. By integrating advanced processing techniques, expanding data sources, and optimizing system performance, this platform can evolve into a comprehensive tool for traders, analysts, and financial institutions. This project has laid a strong foundation, demonstrating the power of real-time data ingestion and modern data architectures in the financial domain.