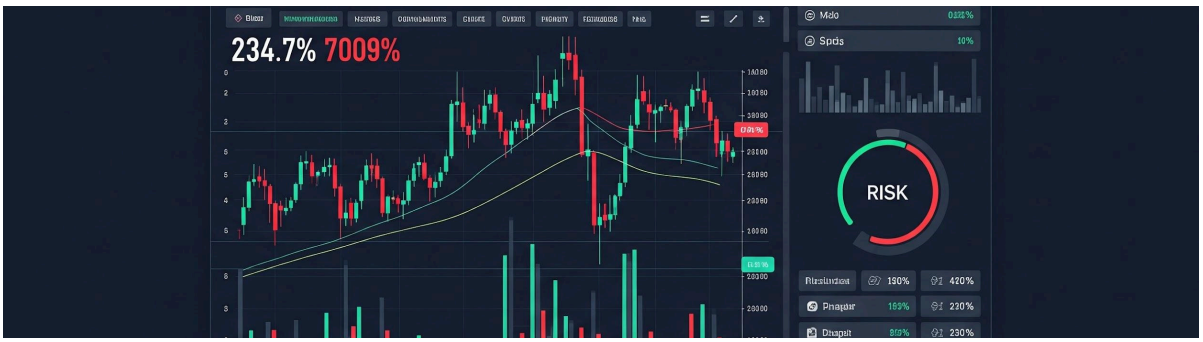


Real-Time Stock Risk Analytics Platform

BY-Prateek Singh

Abstract

This project report details the development of an end-to-end data analytics system designed for real-time monitoring and risk assessment of the US stock market. The primary objective was to build a robust platform capable of ingesting live stock data, processing key risk metrics, and visualizing actionable insights through a Business Intelligence (BI) dashboard. The system employs a modern data architecture utilizing a Python-based producer/consumer model, **RabbitMQ** for event streaming, and **MySQL** for data persistence. Critical risk metrics, including volatility and a binary **HIGH/LOW** risk classification, are calculated in real-time. The final layer is a **Power BI** dashboard that provides interactive, up-to-the-second views of stock performance, volatility, and system health. This platform serves as a proof-of-concept for high-frequency data processing and practical financial risk analytics.

1. Introduction

1.1 Project Objective

The core objective of this project was to construct a functional, real-time stock monitoring and risk analytics platform. The system is designed to provide timely and reliable risk indicators, moving beyond static, end-of-day reporting to enable immediate, data-driven decision-making for trading and risk assessment.

1.2 Data Source

The project relies on the **Finnhub API** for fetching real-time US stock prices. This commercial API provides the necessary high-frequency data streams required for a true "real-time" analytics solution.

2. System Architecture and Implementation

The platform is structured as an event-driven microservices architecture, ensuring scalability and decoupling of components. A diagram illustrating the system flow is provided below:

[A block diagram illustrating the data flow from the Finnhub API (Producer) through RabbitMQ (Message Broker) to the Python Consumer (Risk Calculation) and finally to MySQL (Database) and Power BI (Visualization)]

2.1 Backend Architecture

The backend utilizes Python for data ingestion and processing, with RabbitMQ serving as the central message broker.

Component	Technology	Role
Data Producer	Python	Fetches live stock prices from Finnhub API and publishes them to RabbitMQ.
Message Broker	RabbitMQ	Facilitates asynchronous communication and event streaming between the producer and consumer.
Data Consumer	Python	Subscribes to RabbitMQ, processes raw data, calculates risk metrics, and loads processed data into MySQL.
Database	MySQL	Persistent storage for raw stock prices, calculated volatility, risk level, and processing timestamps.

2.2 Database Schema

The MySQL database is designed to store the processed, time-series data required for BI visualization.

Column Name	Data Type	Description
`timestamp`	DATETIME	The time of the data point, used for trend analysis.
`symbol`	VARCHAR	The stock ticker symbol (e.g., 'AAPL', 'MSFT').
`price`	DECIMAL	The current intraday stock price.
`volatility`	DECIMAL	The calculated volatility metric.
`risk_level`	VARCHAR	Risk classification ('HIGH' or 'LOW').

3. Risk Analytics Methodology

The platform focuses on volatility as the primary indicator of risk, calculated and classified in the consumer application.

3.1 Volatility Calculation

Volatility is calculated using a short-term moving standard deviation of the stock's price, providing a measure of price dispersion over a recent window of time. The formula used is:

$$\sigma_t = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_{t-i} - \mu)^2}$$

Where:

- σ_t = **Volatility at time** t
- N = **Window size** (number of past observations)
- P_{t-i} = **Stock price at time** $t - i$
- μ = **Mean price over the window**, calculated as:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} P_{t-i}$$

3.2 Risk Classification

The calculated volatility is classified into two discrete risk levels:

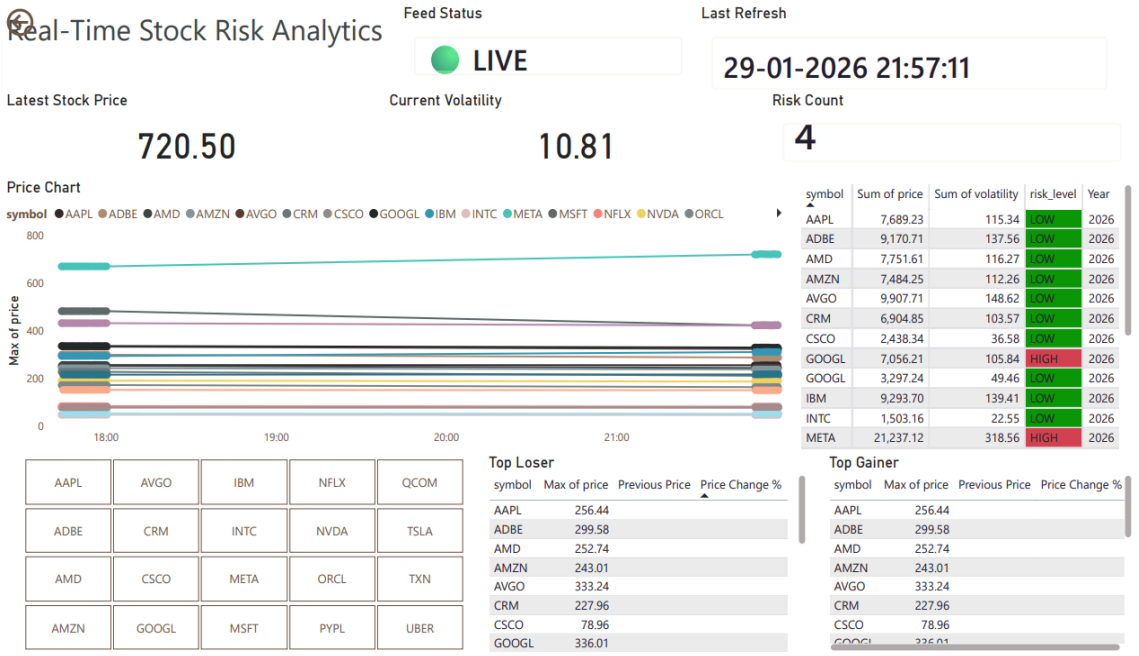
- **HIGH:** Volatility exceeds a predetermined threshold (e.g., above the 80th percentile of historical volatility).
- **LOW:** Volatility is below the predetermined threshold.

3.3 Key Performance Indicator (KPI)

A key operational metric calculated is the **Risk Count KPI**, which represents the number of stocks currently classified as **HIGH** risk within the monitoring universe. This provides an aggregate view of market instability.

4. Visualization and Business Intelligence

The final output of the project is a professional Power BI dashboard that connects directly to the MySQL database to visualize the real-time insights.



4.1 Power BI Dashboard Components

The dashboard is structured to provide both high-level summaries and detailed trend analysis:

- **Intraday Price Trend:** A line chart showing the price movement of the selected stock over the session, driven by the `timestamp` field.
- **Latest Stock Price KPI:** The most recent price for the selected symbol.
- **Current Volatility KPI:** The calculated volatility for the selected symbol.
- **Risk Count KPI:** The total number of **HIGH** risk stocks.
- **Top Gainers and Top Losers:** A table view showing the stocks with the largest percentage price changes.
- **Live Feed Status:** A dynamic indicator showing if the data feed is **LIVE** or **DELAYED**.
- **Interactive Symbol Slicers:** Allows users to filter the dashboard by stock symbol.

4.2 DAX Measures for Advanced Analysis

Several Data Analysis Expressions (DAX) were created in Power BI to enhance the analytical capability of the dashboard:

DAX Measure	Description
`Previous Price`	Calculates the price of the stock at the previous timestamp.

DAX Measure	Description
`Price Change Percentage`	Measures the percentage difference between the current price and a baseline (e.g., the previous price).
`Last Updated Timestamp`	Displays the most recent `timestamp` in the dataset, confirming data recency.
`Data Status Indicator`	Compares `Last Updated Timestamp` to the current time to determine **LIVE** / **DELAYED** status.

5. Challenges and Solutions

The implementation of a real-time system presented several technical challenges that required specific solutions:

Challenge	Solution Implemented
Finnhub API Limits	Implemented a rate-limiting mechanism and intelligent batching on the Python producer to manage request frequency.
Aggregation Issues (Volatility)	Ensured a sliding-window mechanism was correctly implemented in the consumer logic to calculate moving standard deviation over the correct time intervals.
DAX Filter Errors	Utilized the `ALL()` and `ALLEXCEPT()` functions in DAX to correctly manage filter contexts across different visuals (e.g., calculating `Risk Count KPI` regardless of the symbol slicer).
Dashboard Layout Optimization	Adopted a mobile-first design approach and used responsive Power BI containers to ensure a clean, intuitive layout for complex data.

6. Use Cases

The developed platform has broad applicability in the financial domain:

- **Real-Time Market Monitoring:** Provides traders with an immediate view of price and risk fluctuations.
- **Risk Assessment:** Offers a quantitative, dynamic risk classification for portfolio management.
- **Trading Analytics:** Serves as a data source for developing and back-testing high-frequency trading strategies.
- **BI Dashboards:** A template for integrating live, streaming data into a professional Business Intelligence environment.

7. Future Enhancements

To evolve the platform into a production-ready system, the following enhancements are planned:

- **WebSocket Streaming:** Migrate from HTTP API polling to a more efficient **WebSocket** connection for lower-latency data ingestion.
- **Advanced Risk Metrics:** Incorporate more sophisticated risk measures such as **Value-at-Risk (VaR)**, **Conditional Value-at-Risk (CVaR)**, and the **Sharpe Ratio**.
- **Alerts and Notifications:** Implement a feature in the consumer to trigger email or SMS alerts when a stock's risk level transitions to **HIGH**.
- **Cloud Deployment:** Transition the architecture from local deployment to a scalable cloud environment (e.g., AWS, Azure) using containerization (Docker/Kubernetes).

Conclusion

The "Real-Time Stock Risk Analytics Platform" successfully demonstrates the integration of modern data engineering principles with financial analytics. Technically, this project provided deep learning in managing asynchronous event streams using **RabbitMQ**, optimizing database performance for high-frequency writes (**MySQL**), and mastering advanced visualization techniques (**Power BI** and **DAX**). The outcome is a functional, end-to-end system capable of transforming raw, real-time market data into actionable risk insights. This platform is a testament to the power of a unified data pipeline in solving complex, time-sensitive business problems. The successful implementation of volatility-based risk classification and its immediate visualization provides a solid foundation for future development into a commercial-grade trading and risk management tool.