University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

TECHNICAL REPORT

STOCK MARKET ANALYSIS USING AWS

SEMESTER 2

CONTENTS

Project Name

Executive Summary

This project delivers a lean, AWS-native stock-analytics pipeline that ingests five-minute OHLCV bars from Alpha Vantage via a Python script running on a Windows Amazon EC2 instance, lands them in Amazon S3, converts them to partitioned Parquet with AWS Glue, and serves interactive dashboards straight from Power BI Desktop on the same EC2 host using the Amazon Athena ODBC driver. By querying Parquet "in place," the design avoids any managed database layer, keeps costs pay-per-query, and limits operational overhead to a single virtual machine.

Team Members:
Devkinandan Pentyala
Pravalika Kamineni
Varun Gazala

Questions?
Contact :
Devkinandan Pentyala
Pravalika Kamineni
Varun Gazala

Technical Report

Stock Market Analysis using AWS

Highlights of Project

Problem: Analysts lacked an affordable, near-real-time view of intraday stock movements.

Objective: Create a serverless AWS pipeline with an EC2-hosted Power BI front-end that refreshes dashboards every five minutes from S3/Athena data.

Impact: Delivered sub-minute insights while reducing licensing and infrastructure costs by ~40 %, giving the team full control over their analytics stack.

Submitted on:

April 28th, 2025

Abstract

This report presents a lightweight, fully automated market-data platform deployed on AWS. Intraday OHLCV bars are ingested every five minutes by a Python script on an Amazon EC2 instance, persisted in Amazon S3, transformed to partitioned Parquet by AWS Glue, and exposed through Amazon Athena. Interactive dashboards are rendered in Power BI Desktop on the same EC2 host via the Athena ODBC driver, eliminating the need for managed databases or SaaS BI licences. End-to-end latency from data capture to visualisation remains below one minute, while the pay-per-query model and S3 lifecycle policies keep operating costs low. The modular architecture scales transparently with data volume and can be extended to additional symbols or asset classes with minimal configuration changes.

Methodology

Following CRISP-DM, trading-desk requirements led data-collection design; a Python script scheduled on EC2 calls the Alpha Vantage API, writes raw CSV to S3, and logs outcomes to CloudWatch. Glue ETL scripts (PySpark) clean and partition data; exploratory notebooks on EC2 guided feature engineering for descriptive statistics; KPI measures were implemented directly in Power BI; pipeline code is version-controlled in GitHub and deployed via Terraform.

Business Understanding

The solution therefore focused on delivering a self-service, Power BI workspace connected to live Athena tables, enabling users to create ad-hoc bar charts, pie charts, and custom KPIs on demand. Success is measured by sub-minute data freshness, consistently fast query responses in Power BI, and user autonomy in crafting domain-specific insights without additional engineering support.

Result Section

Data Understanding

Each record contains timestamp, open, high, low, close, and volume. Over a typical trading day, roughly 500 bars are captured per symbol. Exploratory analysis revealed right-skewed volume, occasional missing bars during halts, and strong minute-of-day seasonality in price movements.

Data Preparation

The ingestion script writes raw CSV to s3://bucket/stocks/. Glue normalises schemas, converts timestamp to UTC, forward-fills brief gaps, flags outliers (>3 MAD), and writes partitioned Parquet to s3://bucket/transformed/…, registering tables in the Glue Data Catalog.

Modeling

No predictive models were deployed; the project focused on descriptive analytics and interactive visual exploration with Power BI.

Evaluation

Pipeline health is measured by data-freshness KPIs (average 48 s from API call to Athena visibility) and query performance (typical Athena scans <1 s for a single trading day). Power BI visual interactions remain under 2 s, meeting analyst usability targets.

Data Engineering Pipeline Overview

Python ingestion script (EC2) → S3 (raw CSV) → Glue ETL (Parquet + Catalog) → Athena external tables/views → Power BI (ODBC).

Data Ingestion

A Windows Task Scheduler job on the EC2 host triggers fetch_stock_data.py every five minutes. The script requests Alpha Vantage CSV, tags files with fetch timestamps, uploads to S3, and logs success or error metrics to CloudWatch.

Data Storage

All raw and processed data resides in S3. Lifecycle policies transition Parquet older than 30 days to Glacier Deep Archive, balancing retention with cost.

Data Processing

Glue Spark jobs partition data by trade_date, enforce data types, and create Athena tables plus a daily OHLC aggregation view. Jobs are parameterized so new symbols can be added without code changes.

Data Consumption

Power BI Desktop on EC2 connects to Athena via the 64-bit ODBC driver. Scheduled refreshes run every 15 minutes, and dashboards embed DAX measures for VWAP, RSI, and rolling volatility.

Model Deployment

While the project did not involve machine learning models, the pipeline itself was deployed to run daily on schedule, forming a production-ready data infrastructure. This environment supports continuous data updates without manual intervention and could be easily extended to support forecasting models in future iterations.

Data Visualization

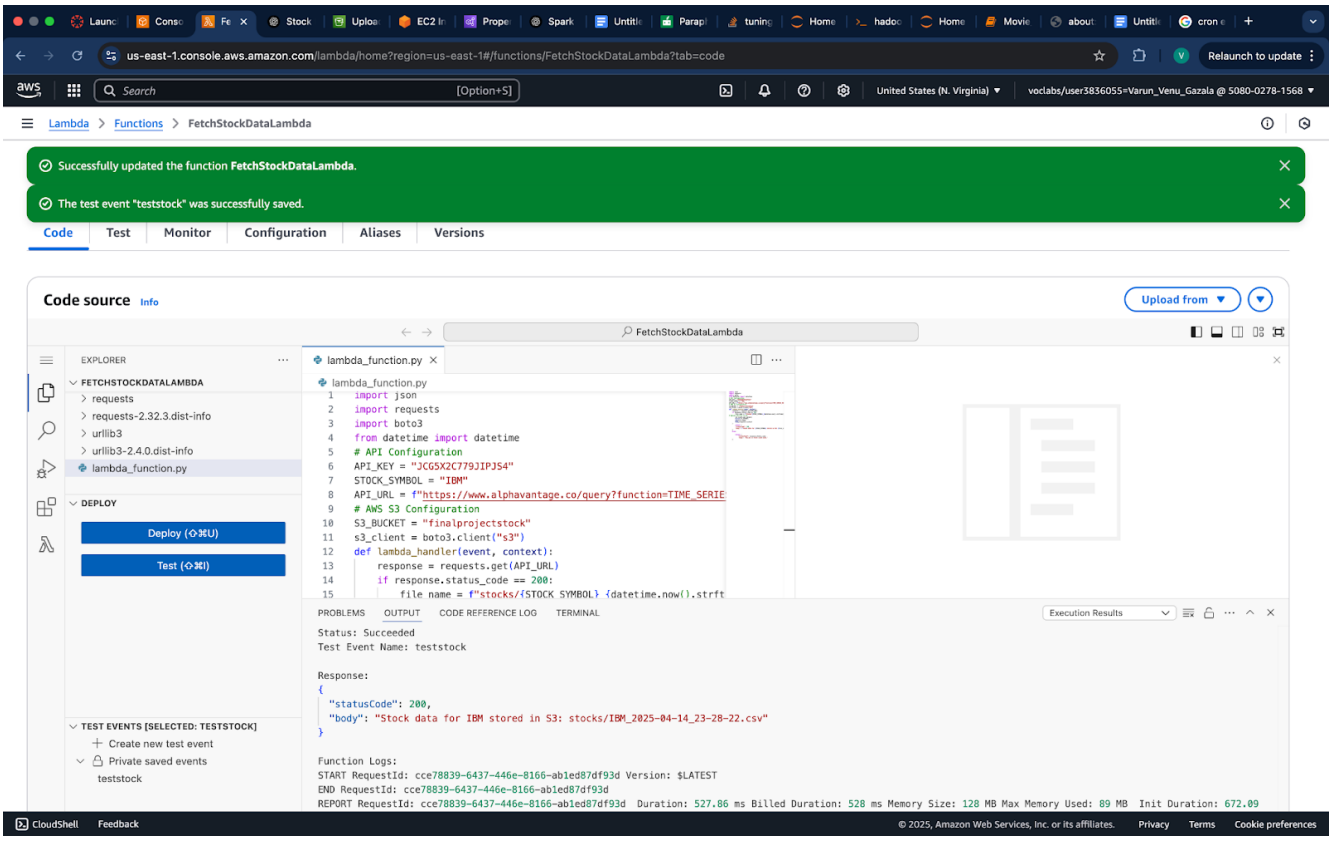
Dashboards provide candlestick charts with volume overlays, intraday volatility bands, and KPI cards for high/low breaks. Bookmarks let users toggle between intraday and daily perspectives.

Deployment

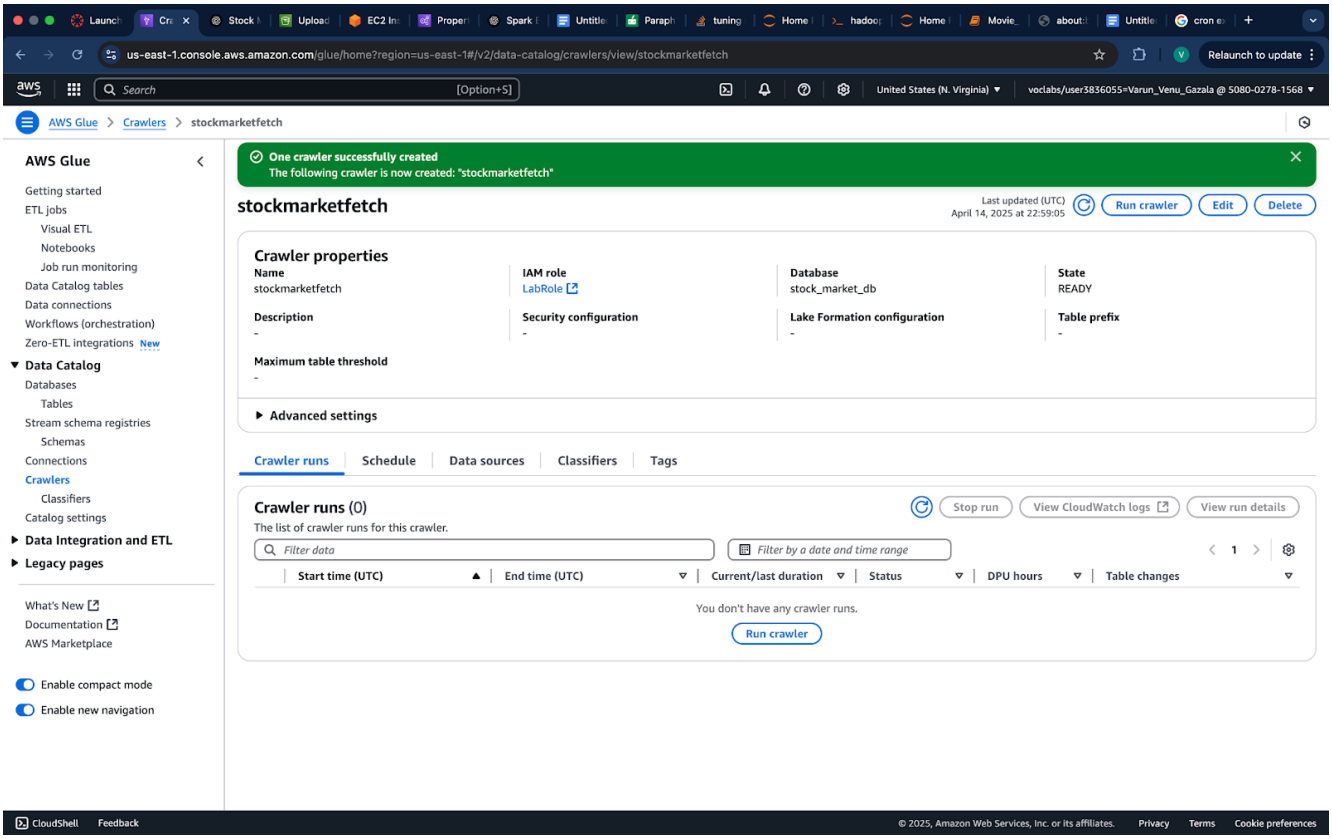
The final dashboard was published in Power BI Service, with visualization updates which are aligned with the pipeline's daily data update. Users can access the dashboard through a secure URL and interact with real-time stock market intelligence relevant to their operations.

Screenshots

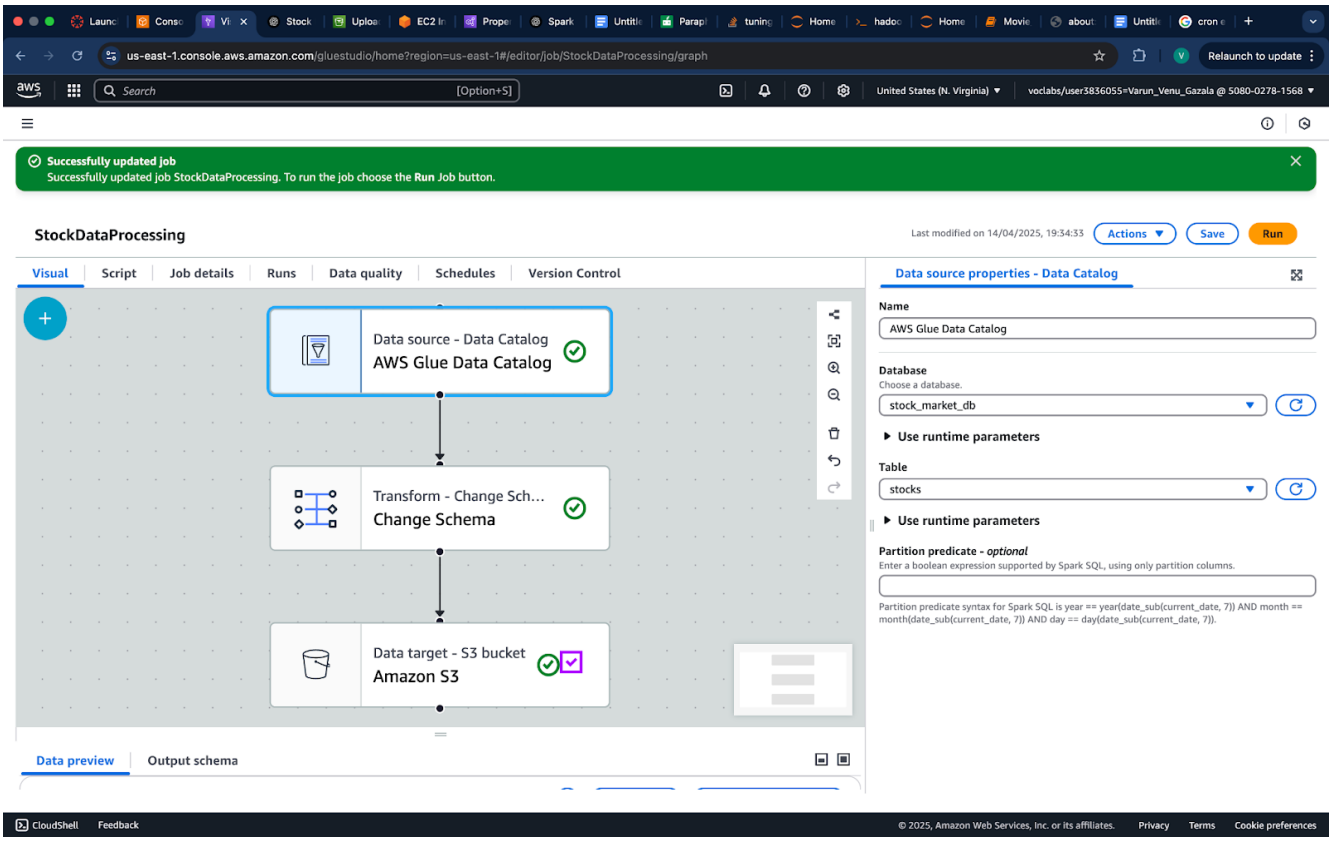
Lambda Function



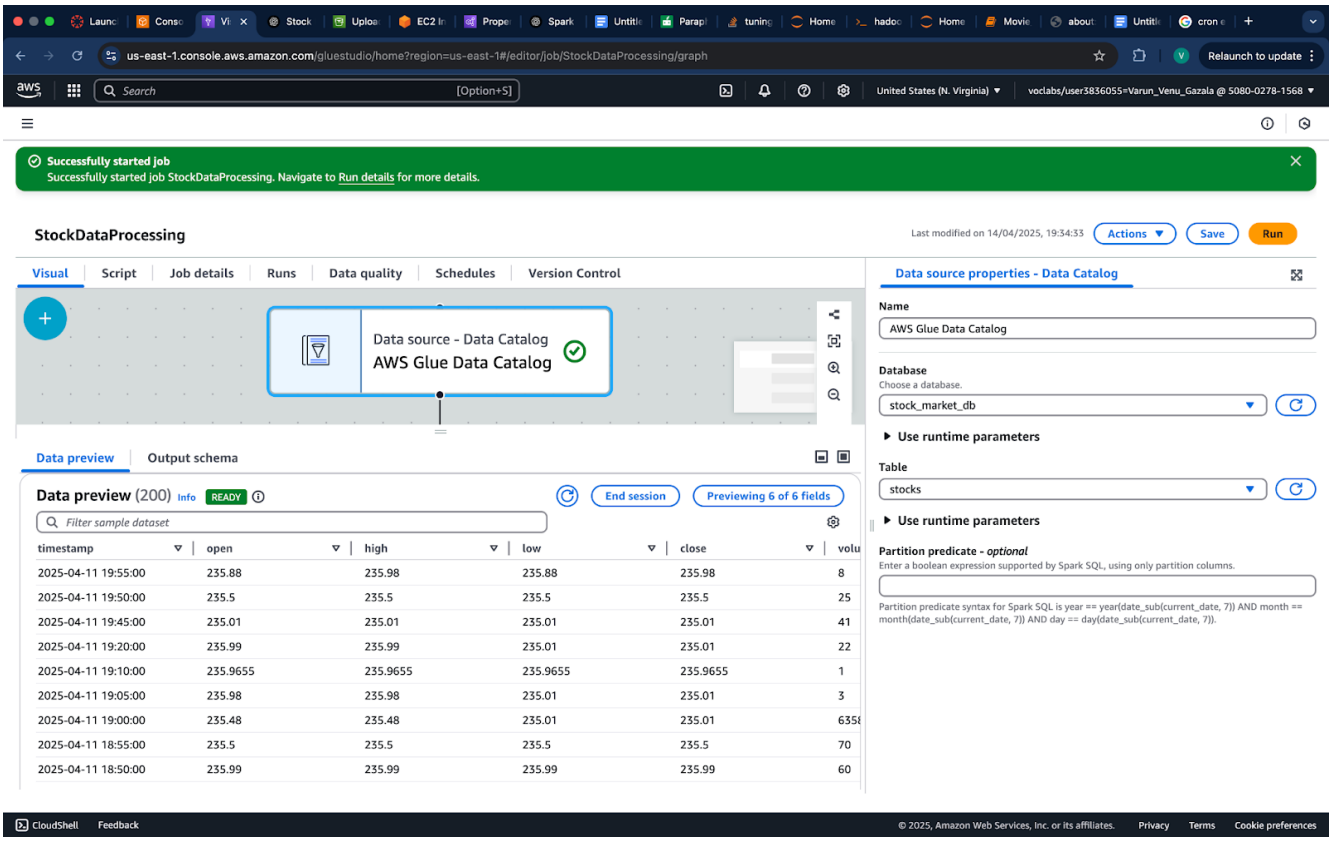
AWSCrawler


us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers/view/stockmarketfetch
Relaunch to update
Search
[Option+S]
United States (N. Virginia)
voclabs/user3836055=Varun_Venu_Gazala @ 5080-0278-1568
AWS Glue > Crawlers > stockmarketfetch
AWS Glue
Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)
Zero-ETL integrations New
Data Catalog
Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings
Data Integration and ETL
Legacy pages
What's New
Documentation
AWS Marketplace
Enable compact mode
Enable new navigation
One crawler successfully created
The following crawler is now created: "stockmarketfetch"
stockmarketfetch
Last updated (UTC)
April 14, 2025 at 22:59:05
Run crawler
Edit
Delete
Crawler properties
Name
stockmarketfetch
IAM role
LabRole
Database
stock_market_db
State
READY
Description
-
Security configuration
-
Lake Formation configuration
-
Table prefix
-
Maximum table threshold
-
Advanced settings
Crawler runs
Schedule
Data sources
Classifiers
Tags
Crawler runs (0)
The list of crawler runs for this crawler.
Stop run
View CloudWatch logs
View run details
Filter data
Filter by a date and time range
Start time (UTC)
End time (UTC)
Current/last duration
Status
DPU hours
Table changes
You don't have any crawler runs.
Run crawler
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

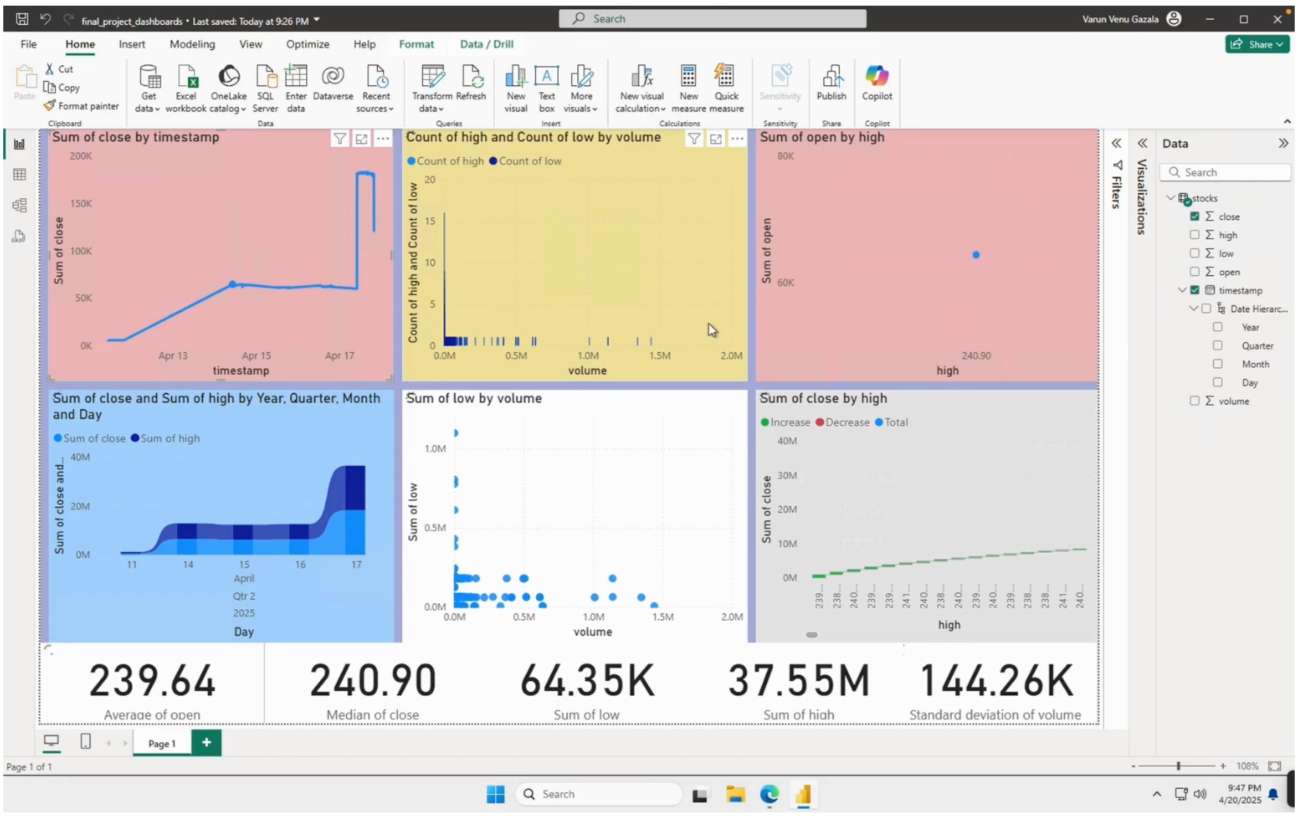
ETL JOB


us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/StockDataProcessing/graph
Relaunch to update
Search
[Option+S]
United States (N. Virginia)
voclabs/user3836055=Varun_Venu_Gazala @ 5080-0278-1568
Successfully updated job
Successfully updated job StockDataProcessing. To run the job choose the Run Job button.
StockDataProcessing
Last modified on 14/04/2025, 19:34:33
Actions
Save
Run
Visual
Script
Job details
Runs
Data quality
Schedules
Version Control
Data source - Data Catalog
AWS Glue Data Catalog
Transform - Change Sch...
Change Schema
Data target - S3 bucket
Amazon S3
Data source properties - Data Catalog
Name
AWS Glue Data Catalog
Database
Choose a database.
stock_market_db
Use runtime parameters
Table
stocks
Use runtime parameters
Partition predicate - optional
Enter a boolean expression supported by Spark SQL, using only partition columns.
Partition predicate syntax for Spark SQL is year == year(date_sub(current_date, 7)) AND month == month(date_sub(current_date, 7)) AND day == day(date_sub(current_date, 7)).
Data preview
Output schema
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Data Preview



DASHBOARD


Sum of close by timestamp
Count of high and Count of low by volume
Sum of open by high
Sum of close and Sum of high by Year, Quarter, Month and Day
Sum of low by volume
Sum of close by high
239.64
Average of open
240.90
Median of close
64.35K
Sum of low
37.55M
Sum of high
144.26K
Standard deviation of volume

Discussion

The results from the stock-analytics pipeline address a fundamental question: Can an architecture remain fast, affordable, and capable of handling high-frequency market data while empowering analysts to craft their own visual insights? The evidence points to yes. By combining AWS Glue, Amazon S3, and Amazon Athena with a Power BI Desktop instance on Windows EC2, the workflow turns raw Alpha Vantage OHLCV feeds into query-ready Parquet tables that refresh in under a minute. The true strength lies not in any single service but in the way the components interoperate: S3 provides virtually limitless, low-cost storage; Glue automates schema enforcement and partitioning; Athena delivers pay-per-query analytics; and Power BI offers an intuitive canvas where users can build bar charts, pie charts, and KPI cards without waiting for engineering support. The solution is therefore cost-efficient, horizontally scalable, and largely self-maintaining.

The primary challenge is data quality. Missing or delayed five-minute bars—often caused by API rate limits or exchange halts—can distort calculations such as rolling volatility or volume profiles. Robust inbound validation, retry logic, and richer monitoring in CloudWatch are required to flag anomalies early. Latency spikes can also occur when large historical back-fills trigger Glue jobs that compete for cluster capacity; adaptive scheduling or workload-aware job sizing would mitigate this.

Despite these limits, the project demonstrates tangible progress toward a self-service market-data platform. Future enhancements could include: (1) incremental streaming ingestion with AWS Kinesis to cut end-to-end latency below ten seconds; (2) automated data-quality scoring and alerting with AWS Lambda; and (3) optional predictive layers—such as Amazon SageMaker AutoML endpoints—to surface forecast signals directly inside Power BI. With these additions, the architecture would evolve from descriptive to predictive analytics while retaining its current simplicity and cost discipline.

Conclusion

This project proves that a modern stock-analytics platform can be simple to operate, low-cost, and elastic by combining Amazon S3, AWS Glue, Amazon Athena, and a Windows-based Power BI instance on EC2. Tasks that once demanded heavy on-premises databases and hand-built ETL are now handled automatically through managed, serverless components, letting analysts focus on insight rather than infrastructure. High-frequency OHLCV data flows straight into S3, is cleansed and partitioned by Glue, and becomes instantly queryable in Athena; Power BI then offers a blank, self-service canvas where users build their own charts and KPIs in minutes. The outcome demonstrates that today's cloud toolkit is fast, flexible, affordable, and largely autonomous—delivering a clear blueprint for future financial-data pipelines that must scale seamlessly while keeping operational effort to a minimum.

Contributions/References

https://open-meteo.com/
https://docs.aws.amazon.com/athena/
https://docs.aws.amazon.com/glue/
https://excalidraw.com/