Stock Analytics Pipeline and API Documentation

Table of Contents

Introduction

The Stock Analytics Pipeline is an end-to-end solution for aggregating, processing, and analyzing financial data. It integrates stock prices, news, and social media discussions into a unified database, providing insights via calculated metrics, sentiment analysis, and trends. Users can access this data through a Flask-based API, which supports dynamic querying and user-friendly endpoints.

Datasets Overview

Back to Top

- **Stock Metrics**:
  - Source: Alpaca and Finnhub APIs.
  - Includes historical prices, trading volume, and calculated metrics like moving averages and volatility.
- **News Articles**:
  - Source: News API.
  - Financial articles related to stocks.
  - Fields include publication date, source, title, and description.
- **Reddit Posts**:
  - Source: Reddit API.
  - Extracts stock-related discussions, enriched with sentiment scores and user engagement metrics (e.g., scores, comments).

Data Transformation Process

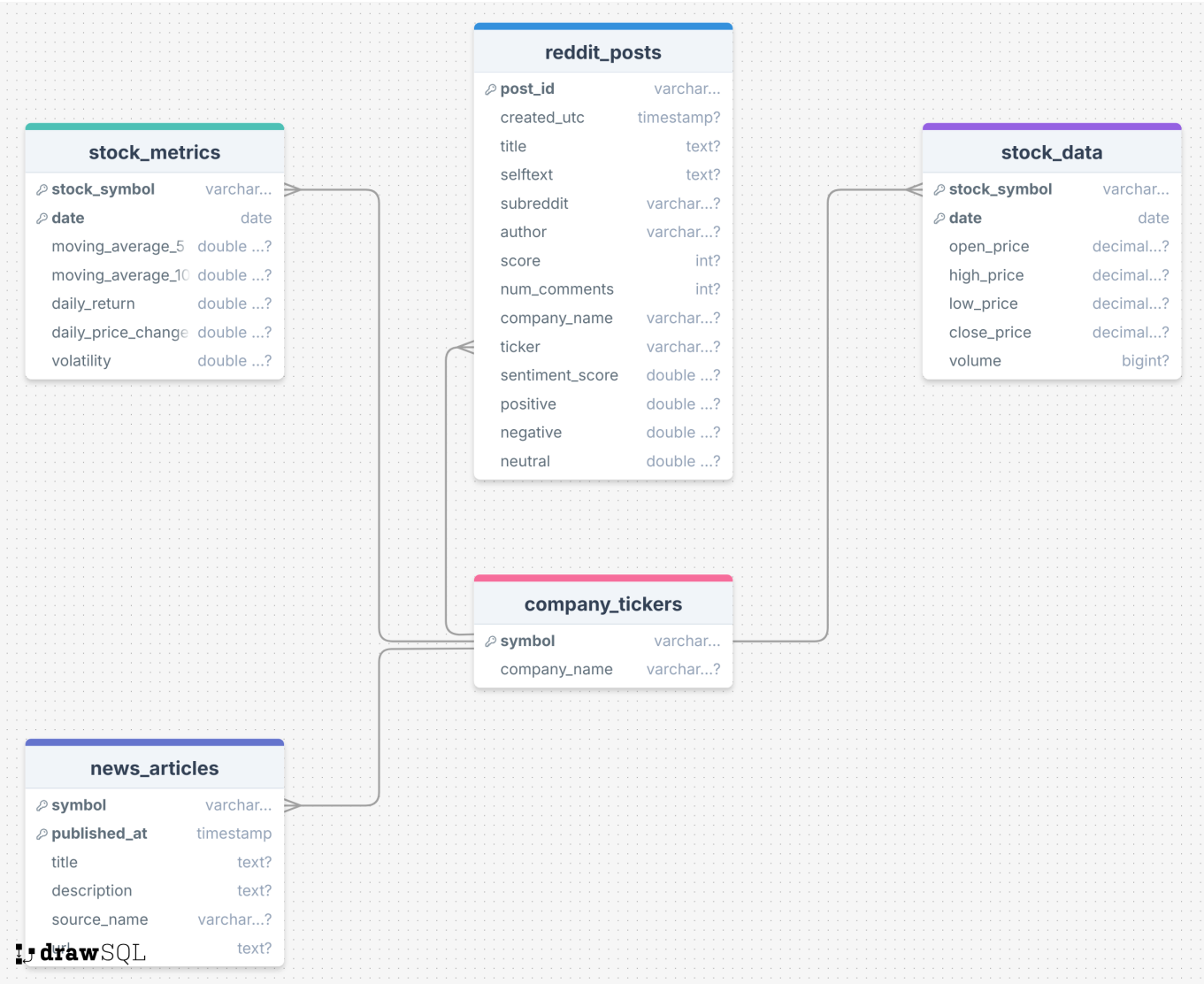
Back to Top

1. **Stock Data**:
    - Extracted using Alpaca/Finnhub APIs.
    - Metrics include 5-day/10-day moving averages, daily returns, and volatility.
    - Stored in the `stock_metrics` table.
2. **News Articles**:
    - Extracted from the News API and filtered for relevance.
    - Transformed to include publication timestamps and metadata.
    - Stored in the `news_articles` table.
3. **Reddit Posts**:
    - Extracted using the Reddit API.
    - Sentiment analysis applied to derive scores.
    - Stored in the `reddit_posts` table.
4. **Orchestration**:
    - Managed via Apache Airflow.
    - Tasks handle extraction, transformation, and loading (ETL).

Entity-Relationship Diagram (ERD) and Database Schema

Back to Top

The database schema forms the backbone of the Stock Analytics Pipeline. The following tables are central to the system, and their relationships are illustrated in the ERD below:



Database Structure:

1. `company_tickers`:
    - **Purpose**: Stores stock ticker symbols and their associated company names.
    - **Columns**:
        - `symbol` (Primary Key): Unique identifier for each stock.
        - `name`: The name of the company.
    - **Relationships**: Referenced by `stock_data`, `stock_metrics`, `news_articles`, and `reddit_posts`.
2. `stock_data`:
    - **Purpose**: Contains raw stock data retrieved from APIs.
    - **Columns**:
        - `stock_symbol` (Foreign Key): Linked to `company_tickers.symbol`.
        - `date`: The trading date.
        - `open_price`, `high_price`, `low_price`, `close_price`: The daily price data.
        - `volume`: The trading volume.
    - **Primary Key**: (`stock_symbol`, `date`).
3. `news_articles`:

- **Purpose**: Stores financial news articles related to stocks.
- **Columns**:
  - `symbol` (Foreign Key): Linked to `company_tickers.symbol`.
  - `published_at`: Publication date and time of the article.
  - `title`: The title of the article.
  - `description`: A brief summary of the article.
  - `source_name`: The source or publisher of the article.
  - `url`: A link to the full article.
- **Primary Key**: (`symbol`, `published_at`).

4. **`reddit_posts`**:

- **Purpose**: Captures Reddit discussions about stocks.
- **Columns**:
  - `post_id` (Primary Key): Unique identifier for each post.
  - `created_utc`: Timestamp of the post.
  - `title`: Title of the post.
  - `selftext`: Post content.
  - `subreddit`, `author`: Metadata about the post's source.
  - `score`, `num_comments`: Engagement metrics.
  - `ticker` (Foreign Key): Linked to `company_tickers.symbol`.
  - `sentiment_score`, `positive`, `negative`, `neutral`: Sentiment analysis results.

5. **`stock_metrics`**:

- **Purpose**: Stores processed analytical metrics for stocks.
- **Columns**:
  - `stock_symbol` (Foreign Key): Linked to `company_tickers.symbol`.
  - `date`: The date for which metrics are calculated.
  - `moving_average_5`, `moving_average_10`: Short-term moving averages.
  - `daily_return`: The percentage change in price.
  - `daily_price_change`: Absolute change in price.
  - `volatility`: A measure of price fluctuation.
- **Primary Key**: (`stock_symbol`, `date`).

Summary of Database Relationships

Back to Top

- The `company_tickers` table acts as the central node, referenced by all other tables.
- Foreign key constraints ensure data consistency and integrity across tables.
- The ERD reflects a modular design, enabling independent data processing for stocks, news, and Reddit discussions, while maintaining connections through the `symbol` key.

Automation

Back to Top

- **Airflow**:
  - DAG-based automation (`data_engineering_pipeline_dag.py`) handles ETL tasks.
  - Configured for daily updates, ensuring near-real-time data availability.
- **Database Initialization**:
  - Executed via `init.sql`, which creates tables and establishes relationships.

API Overview

Back to Top

The Flask API provides access to processed data, allowing users to query and visualize results.

Endpoints:

1. `/api/metrics/<symbol>`: Returns stock metrics (e.g., moving averages, volatility).
2. `/api/news/<symbol>`: Fetches relevant news articles.
3. `/api/reddit/<symbol>`: Fetches Reddit posts with sentiment analysis.
4. `/api/summary/<symbol>`: Combines metrics, news, and Reddit discussions.
5. `/api/sentiment_trend/<symbol>`: Graphs daily sentiment trends.
6. `/api/compare_metrics`: Compares average metrics across multiple stocks.

Instructions for Running the Pipeline

Back to Top

1. **Unzip the Project Files**:
   - Extract the zipped file to a directory of your choice.
2. **Navigate to the Project Directory**:
   - Open a terminal and `cd` into the unzipped directory.
3. **Start Docker Containers**:
   - Run the following command to bring up the necessary services:

     ```
     docker-compose up --build
     ```
4. **Start the Airflow Scheduler**:

   Option 1: Command-Line Instructions

   1. **Unpause the DAG**
      Unpause the DAG so it can run manually or automatically:

```
docker exec -it airflow_jupyter_container airflow dags unpause
data_engineering_pipeline
```

2. **Verify the DAG is Unpaused**
   Confirm the DAG is now active:

   ```
   docker exec -it airflow_jupyter_container airflow dags list
   ```

   Look for `paused` set to `False` for `data_engineering_pipeline`.

3. **Trigger the DAG**
   Start the DAG manually:

   ```
   docker exec -it airflow_jupyter_container airflow dags trigger
   data_engineering_pipeline
   ```

4. **Monitor the DAG Execution**
   Check the status of tasks within the DAG:

   ```
   docker exec -it airflow_jupyter_container airflow tasks list
   data_engineering_pipeline
   ```

   View task logs for debugging:

   ```
   docker exec -it airflow_jupyter_container airflow tasks logs
   data_engineering_pipeline <task_id> <execution_date>
   ```

   Replace `<task_id>` with the task name and `<execution_date>` with the specific run date.

Option 2: Use the Web UI

1. **Access the Web UI**
   Open the Airflow Web UI in your browser at http://localhost:8080.
   Log in with the credentials you set during the Airflow setup (`admin` / `admin` by default).

2. **Unpause the DAG**

   - Go to the **DAGs** tab.
   - Find `data_engineering_pipeline` in the list.
   - Click the toggle switch in the **Active** column to unpause it.

3. **Trigger the DAG**

- Click the play button (▶) on the right-hand side of the DAG row.
- Confirm the DAG run.

4. **Monitor the DAG**

   - Click on the DAG ID (`data_engineering_pipeline`) in the list to open its details.
   - View the **Graph View** or **Tree View** to track task progress.
   - Click individual tasks to see logs and other execution details.

Additional Notes

- If you're unsure of the DAG's status or encounter issues, check the scheduler logs:

```
docker logs airflow_jupyter_container
```

- Ensure the DAG file is error-free and correctly placed in the `airflow/dags/` directory. Logs for parsing errors can help identify issues.

5. **Access the API**:

   - Open a browser and go to http://localhost:5001 to access the API.

   OR

   - For external access, install and set up Ngrok using Homebrew:

     - Open your terminal and run:

       ```
       brew install ngrok/ngrok/ngrok
       ```

     - **Sign Up for a Ngrok Account**:

       - Visit https://ngrok.com/signup and create a free account.
       - After signing up, log in to your Ngrok dashboard and copy your personal authtoken from https://dashboard.ngrok.com/get-started/your-authtoken.

     - **Authenticate Ngrok**:

       - Authenticate Ngrok with your account by running:

       ```
       ngrok config add-authtoken <your-authtoken>
       ```

     - **Run Ngrok**:

       - To expose your Flask API to the internet, start an Ngrok tunnel by running:

       ```
       
       ```

```
ngrok http 5001
```

- Ngrok will provide a public URL (e.g., `https://xyz123.ngrok-free.app`) that can be shared for external access to your API.

Example API Queries

Back to Top

1. **Metrics Query**:

   ```
   GET /api/metrics/AAPL
   ```

   Returns:

   ```
   {
       "date": "2024-11-16",
       "moving_average_5": 180.45,
       "volatility": 1.75,
       ...
   }
   ```

2. **News Query**:

   ```
   GET /api/news/AAPL
   ```

   Returns a list of recent articles with links and timestamps.

3. **Reddit Query**:

   ```
   GET /api/reddit/AAPL
   ```

   Returns:

   - Post titles.
   - Sentiment scores.
   - Engagement metrics.

4. **Summary Query**:

   ```
   GET /api/summary/AAPL
   ```

Consolidates metrics, news, and Reddit data.

5. **Compare Metrics**:

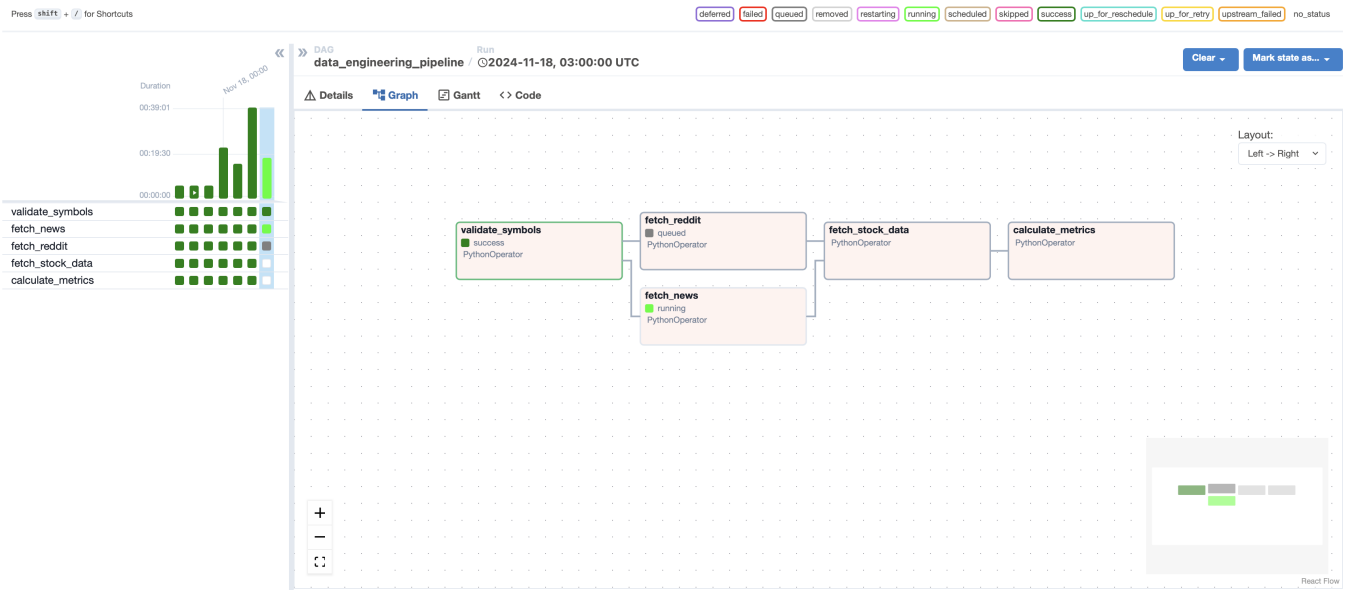
```
POST /api/compare_metrics
Body: { "symbols": ["AAPL", "MSFT"] }
```

Compares averages across stocks.

Screenshots

Back to Top

Airflow Dashboard:



Flask API Home Page:

Welcome to the Stock Analytics API

Explore the following endpoints:

- /api/metrics/<symbol>: Get stock metrics for the given symbol
- /api/news/<symbol>: Get news for the given symbol
- /api/reddit/<symbol>: Get Reddit posts for the given symbol
- /api/summary/<symbol>: Get a summary of metrics, news, and Reddit posts for the given symbol
- /api/sentiment_trend/<symbol>: Get sentiment trend for the given symbol
- /api/compare_metrics: Compare stock metrics across multiple symbols (query parameter: symbols)

Search for a symbol: Enter symbol (e.g., AAPL) Search

Available Tickers

Symbol	Name
AAPL	Apple Inc
MSFT	Microsoft Corp
NVDA	NVIDIA Corp
GOOGL	Alphabet Inc
AMZN	Amazon.com Inc
TSLA	Tesla Inc
META	Meta Platforms Inc
BRK.B	Berkshire Hathaway Inc
TSM	Taiwan Semiconductor Manufacturing Co Ltd
LLY	Eli Lilly and Co
AVGO	Broadcom Inc
JPM	JPMorgan Chase & Co
NVO	Novo Nordisk A/S
WMT	Walmart Inc
UNH	Unitedhealth Group Inc
XOM	Exxon Mobil Corp
V	Visa Inc
MA	Mastercard Inc
PG	Procter & Gamble Co
ORCL	Oracle Corp
ASML	ASML Holding NV
SHEL	Shell PLC
KO	Coca-Cola Co
PEP	PepsiCo Inc
CSCO	Cisco Systems Inc

Metrics Results example:

Reddit Posts for AAPL

Created At	Title	Score	Comments
2024-10-01 04:54:10	MSFT or AAPL	0	82
2024-05-08 14:07:32	Are you worried about AAPL long term?	347	455
2024-03-21 11:32:28	The U.S. Department of Justice (DOJ) is preparing to sue Apple (AAPL.O), as soon as Thursday for allegedly violating antitrust laws	605	212
2023-09-07 13:15:44	China plans to expand its ban on iPhones, AAPL drops 3%+	773	399
2020-10-13 09:19:24	I have $213,000 in Apple (AAPL) stock, should I start to diversify?	2460	727

Go Back
Back to Home

Key Decisions

Back to Top

1. PostgreSQL chosen for its scalability.
2. Airflow used for task orchestration.
3. Flask selected for its flexibility in building APIs and HTML friendly.

Conclusion & Contacts

Back to Top

This project integrates stock data, news, and social media into a comprehensive analytics platform. It ensures accurate, real-time insights for financial analysis and decision-making. Contact for questions: ahakobi1@jh.edu or nbahou1@jh.edu.