1. Data Extraction (E)

Sources of Data:

MarketStack API: Fetches historical stock data based on specified tickers.

Kaggle Dataset: Loads stock price data from Kaggle's daily-updating stock dataset.

Local CSV File: Reads data from a local CSV file containing stock data.

MongoDB: Fetches stock data from a MongoDB collection.

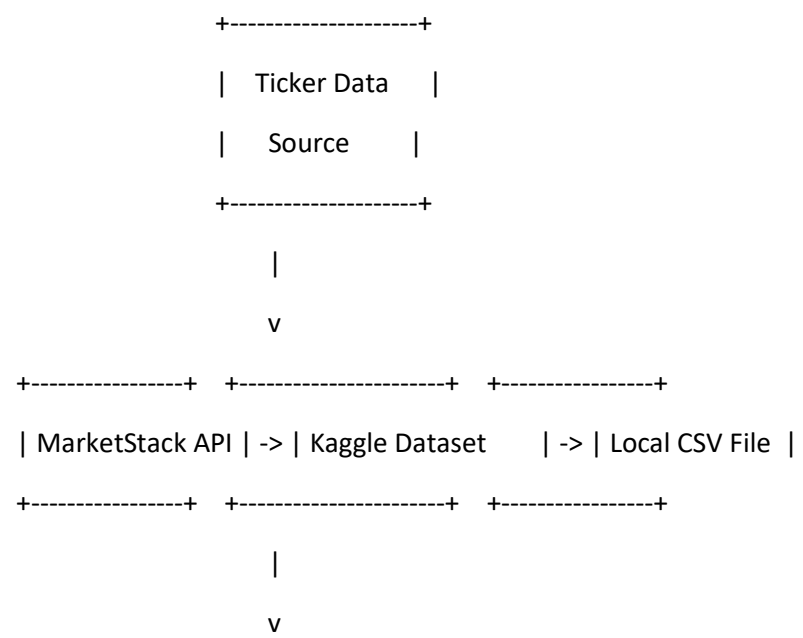
GitHub: Reads stock data from a CSV hosted on GitHub.

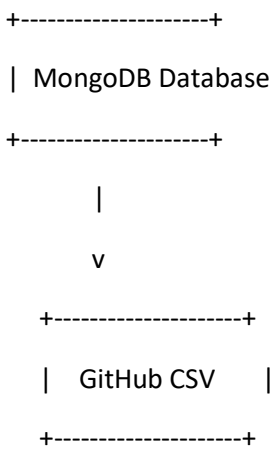
Diagram:

plaintext

Copy

Edit






2. Data Transformation (T)

After extracting data from various sources, the pipeline performs several transformations on the datasets:

Standardization of Timestamps: The date columns across different datasets are standardized to ensure uniformity. This is important for merging the data later.

Normalization of Column Names: The column names are normalized to lowercase and underscores to ensure consistency across datasets.

Handling Missing Values: Missing values are handled using the handle_missing_values method, which adds a new feature (capital gains) in the Kaggle dataset and deals with missing data for other datasets.

Data Validation: Negative values in numerical columns (such as stock prices) are dropped, ensuring only valid financial data is retained.

Feature Engineering: New features are created, such as capital_gains, daily_return, and volatility. This enhances the data for analysis and reporting.

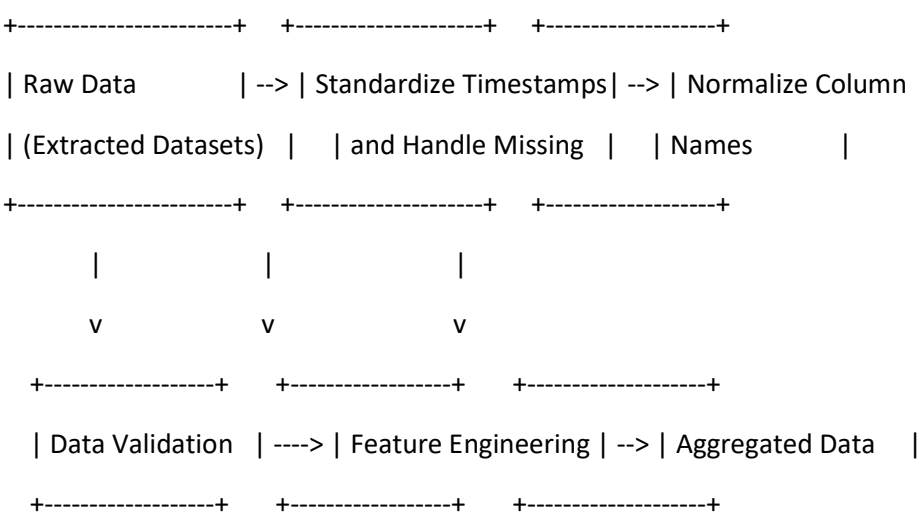
Aggregation: Data is aggregated at a daily level by symbol (ticker) and date, calculating metrics like mean, max, sum, etc.

Diagram:

plaintext

Copy

Edit




3. Data Load (L)

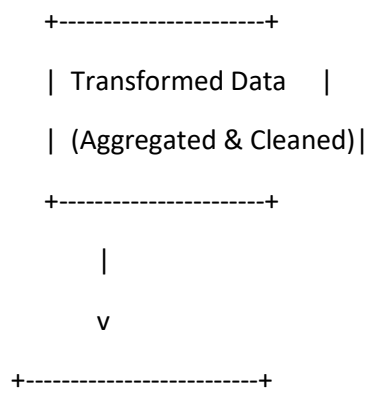
Once the transformation is completed, the final dataset is merged and loaded into MongoDB for further analysis or querying. The dataset is inserted as records in MongoDB.

Diagram:

plaintext

Copy

Edit



| Load Data to MongoDB |

+-------------------------+

Technology/Tool Choices Justification

1. Python:

Why Python: Python is a versatile language with a rich ecosystem of libraries for data processing, making it an ideal choice for building ETL pipelines. Libraries like requests, pandas, numpy, and pymongo provide powerful tools to handle API requests, data transformation, and interaction with databases.

Data Processing: Libraries like pandas allow for easy manipulation, transformation, and aggregation of large datasets.

API Integration: requests allows easy communication with external APIs like MarketStack.

Database Connectivity: pymongo helps interact with MongoDB efficiently.

2. Scheduling Library:

Why Use a Scheduling Library (e.g., Airflow, Luigi, or schedule):

Automatic Data Refresh: Data can be automatically pulled from sources at scheduled intervals, ensuring the pipeline runs in an automated, non-interactive manner.

Error Handling and Monitoring: Scheduling frameworks offer built-in error handling, retry mechanisms, and alerting, which reduces manual intervention in case of failures.

Advantages:

Reduces Manual Errors: Automating the data extraction, transformation, and loading processes minimizes the possibility of human error, such as missing data or incorrect data handling.

Facilitates Rapid Feedback Loops: Automated ETL pipelines allow for faster iterations. As new data becomes available, it can be quickly processed and analyzed without manual steps.

3. MongoDB:

Why MongoDB: MongoDB is chosen due to its flexibility and scalability in handling large amounts of unstructured data. It is particularly well-suited for storing financial stock data, as it can easily accommodate complex documents and evolving datasets.

Scalability: MongoDB can handle large volumes of data and allows for flexible data models, making it ideal for dynamic financial datasets.

Querying and Aggregation: MongoDB provides powerful aggregation features, which are useful for aggregating stock data and running analytics queries.

Pipeline Benefits

Reduces Manual Errors:

The entire pipeline is automated, from data extraction to data load. This reduces the risk of errors from manual data handling or processing.

Facilitates Rapid Feedback Loops:

Once the pipeline is set up and scheduled, data is constantly refreshed and transformed automatically. This leads to rapid iterations and allows the business or team to act on fresh data more efficiently.

Improves Data Integrity through Automated Testing:

With automatic validation and data checks (such as handling missing values, validating negative values, and normalizing columns), data integrity is maintained throughout the pipeline. Automated tests and logging also ensure any issues are quickly identified.

Accelerates Development and Deployment Cycles:

By automating the extraction, transformation, and loading processes, the time needed for data preparation is greatly reduced, accelerating both development and deployment. Teams can focus on analysis and insights, rather than worrying about data cleaning and preparation.

Full Pipeline Overview:

The ETL pipeline you've created is an automated, scalable, and repeatable process that integrates data from multiple sources, processes and transforms it, and loads it into MongoDB for analysis or visualization. By using Python, scheduling libraries, and MongoDB, the pipeline ensures efficiency, scalability, and reduces the complexity of manual data preparation. The use of automated transformations and validations ensures the data is accurate and consistent, ultimately leading to better insights and decisions.