Name of Student:	James Bobby Kiawu
Submitted To:	Ratinder Rajpal, Lecturer.
Willis College:	Introduction to Artificial Intelligence
Intro AI	**Data Cleaning and Preparation**
Date:	November 20, 2025

Technical Report: Data Cleaning, Integration, and Feature Engineering for Stock Price Analysis

Executive Summary

This project focused on preparing a large-scale historical stock price dataset for predictive modeling. The workflow involved cleaning both price and metadata files, merging them into a unified dataset, generating advanced financial indicators through feature engineering, and performing a chronological time-series dataset split. Due to the extremely large dataset (20+ million rows), special attention was paid to memory optimization and efficient file formats. Ultimately, the final cleaned and feature-engineered dataset was saved in Parquet format and split into training, validation, and test sets, making it fully ready for future analytics and modeling tasks.

The project demonstrated strong data management principles, including data integrity validation, handling missing values, efficient computation, and preventing data leakage—essential skills for real-world financial data analysis. Despite technical challenges due to dataset size and Colab RAM limitations, the final workflow is robust, reproducible, and optimized for further machine learning experimentation.

1. Introduction

Financial datasets often contain millions of rows and must be cleaned, standardized, and properly structured before modeling. This project prepares two separate datasets:

1. **Historical Stock Prices** (OHLCV data for multiple tickers across several decades)
2. **Metadata** (ticker-level attributes such as exchange, sector, and industry)

The goal was to:

- Clean both datasets thoroughly
- Ensure high data integrity
- Merge them into a single coherent dataset
- Engineer predictive financial features
- Prepare datasets for time-series forecasting models

This report outlines all technical steps, decisions, justifications, and results.

2. Data Loading

Two files were loaded:

- historical_stock_prices.csv
- historical_stocks.csv

Both were imported into pandas DataFrames for processing. Because the dataset size exceeded typical RAM limits, steps were run iteratively with memory clean-up using gc.collect() to avoid session crashes.

3. Metadata Cleaning

Metadata was cleaned through the following steps:

3.1 Handling Missing Values

- Removed rows with missing **ticker** (primary key)
- Cleaned text-based fields (sector, industry) by:

    - Stripping whitespace
    - Converting to uppercase

3.2 Filling Missing Text Values
Missing entries were replaced with "UNKNOWN", ensuring consistency.

3.3 Deduplication
Duplicate tickers were removed, preserving the first valid entry.

3.4 Reducing to Essential Columns
Only the following were kept:

- ticker
- exchange
- sector
- industry

This reduced metadata size and improved merge speed.

---

4. Stock Price Cleaning

A similar cleaning pipeline was applied to the stock price dataset.

4.1 Data Type Conversion

- Converted date to datetime
- Ensured numeric columns were properly typed

4.2 Missing Price Interpolation

A **per-ticker interpolation** strategy was used: df = df.groupby('ticker').apply(fast_interpolation)
This preserved price continuity while avoiding artificial trends across tickers.

4.3 Index Resetting & Sorting

- Dataset sorted by ticker and date
- Index was reset to prepare for merging

The price dataset was left with no missing OHLCV values.

---

5. Dataset Integration
The cleaned price dataset and cleaned metadata were merged using:
merged_df = hist_df_cleaned.merge(meta_df, on='ticker', how='left')

A **left join** ensured all stock price rows were preserved.
The resulting dataset included:

- OHLCV price columns
- Company metadata
- Date/time index
- Ticker identifier

This integration formed the foundation for feature engineering.

---

6. Feature Engineering
Several powerful market indicators were created.

6.1 Moving Averages
Purpose: track smoothed price trends.

Features created:

- MA_7
- MA_14
- MA_21

These were computed per ticker to maintain independence.

6.2 Volatility Indicators

Purpose: measure fluctuation intensity.
Computed rolling standard deviations:

- STD_7
- STD_14
- STD_30

Higher values indicate increased volatility.

6.3 Price Momentum

The momentum feature calculated the difference between today's price and the price 7 days earlier. Highly predictive for short-term direction signaling.

6.4 RSI (Relative Strength Index)

A 14-period RSI was generated:

- Calculated price deltas
- Extracted average gains & losses
- RSI formula applied.

RSI helps detect overbought/oversold conditions.

6.5 MACD

The MACD indicator was computed using:

- **12-day EMA (fast)**
- **26-day EMA (slow)**
- **MACD line = EMA12 - EMA26**
- **Signal line = 9-day EMA of MACD**

Temporary columns (EMA12, EMA26) were removed to reduce memory usage.

---

7. Dropping Metadata Columns (this step was optional and I took the liberty to drop the cols)

Because categorical modeling was not required for this assignment, the following were removed:

- exchange
- sector
- industry

Purpose:

- Reduce memory usage
- Avoid excessive RAM consumption in Colab

---

8. Saving the Final Dataset

The final dataset was saved as: merged_df_FE.parquet
Why Parquet?

- Smaller file size
- Faster I/O
- Preserves data types

- Supports big-data workflows

The file was downloaded to prevent loss after Colab resets.

---

9. Reloading and Time-Series Splitting

To validate stability, the parquet file was reloaded.

A chronological split was applied:

- **70% Training**
- **15% Validation**
- **15% Test**

Time order was preserved by disabling shuffling.

The following files were created:

- train_df.parquet
- val_df.parquet
- test_df.parquet

These were also downloaded for future modeling.

---

10. Visualizations (Suggested for Final Report)

Please refer to the Google Colab section that contains the various graphs.

---

11. Challenges Encountered

11.1 Colab RAM Limitations

The dataset (20M+ rows) repeatedly caused:

- RAM crashes – this was so frustrating and it took me more than 7hrs to develop a new approach.
- Session resets
- Runtime disconnects

Solution:

Saved intermediate outputs to Parquet, reloaded the dataset fresh, and cleared RAM using gc.collect().

11.2 Merge Crashes

Initial merges failed due to:

- Large DF memory consumption
- Temporary rolling columns

Solution:

Performed merge earlier before heavy feature engineering, then optimized computations.

11.3 Rolling Calculations Were Expensive

RSI, MACD, and MA computations on millions of rows were slow.

Solution:

Used efficient grouped transforms and deleted temporary columns immediately.

11.4 Loss of Uploaded Files After Timeout

Colab clears /content after disconnection.

Solution:

Downloaded parquet files after every major step.

---

12. Lessons Learned

12.1 Memory Optimization Is Critical.
Large datasets require:

- Efficient file formats (Parquet > CSV)
- Minimal temporary variables
- Frequent garbage collection
-

12.2 Always Split Time-Series Data Chronologically
Shuffling creates data leakage.
Preserving time order ensures realistic model evaluation.

12.3 Intermediate Saving Prevents Data Loss
Saving interim parquet files protects progress during long projects.

12.4 Feature Engineering Must Be Grouped by Ticker
Wrong grouping can introduce cross-ticker contamination.

12.5 Documentation Matters
Writing detailed notes is essential for reproducibility—especially with multi-step workflows.

---

13. Conclusion

This project successfully transformed raw stock price and metadata files into a fully cleaned, merged, and feature-engineered dataset suitable for machine learning modeling. Despite challenges with scale and computational limits, the final workflow is optimized, stable, and replicable. The resulting train/validation/test splits can be directly used for forecasting or classification tasks related to market behavior.
This foundation enables the next phases:

- Model development
- Hyperparameter tuning
- Forecasting experiments
- Performance evaluation

A clean dataset is the most important prerequisite—and this project achieved it.