Process Documentation: Finomatrix Stock Data Analysis

1. Introduction
This document provides a step-by-step guide to the stock data analysis project using the Yahoo Finance static API. The project involves fetching historical stock data, performing data cleaning and analysis, and visualizing trends using Python libraries.

2. Objectives
- Retrieve historical stock data using the Yahoo Finance API.
- Perform data preprocessing and exploratory data analysis (EDA).
- Generate key insights and trends from stock data.
- Visualize stock performance using Python.

3. Technology Stack
- Data Retrieval: Yahoo Finance API (yfinance library)
- Programming Language: Python
- Libraries Used: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, Plotly
- Development Environment: Jupyter Notebook

4. Step-by-Step Workflow

Step 1: Data Ingestion
- Install and import the required libraries:

```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- Fetch stock data using the Yahoo Finance API:

```python
stock_symbol = 'AAPL'
stock_data = yf.download(stock_symbol, start='2020-01-01', end='2025-01-01')
```

- Display the first few rows:

```python
  stock_data.head()
  ```

Step 2: Data Preprocessing
- Check for missing values:

  ```python
  stock_data.isnull().sum()
  ```

- Fill or remove missing values if necessary:

  ```python
  stock_data.fillna(method='ffill', inplace=True)
  ```

- Convert the date column to datetime format:

  ```python
  stock_data.index = pd.to_datetime(stock_data.index)
  ```

Step 3: Exploratory Data Analysis (EDA)
- Plot stock price trends:

  ```python
  plt.figure(figsize=(12,6))
  plt.plot(stock_data['Close'], label='Closing Price')
  plt.title(f'{stock_symbol} Stock Closing Price')
  plt.legend()
  plt.show()
  ```

- Calculate moving averages:

  ```python
  stock_data['50_MA'] = stock_data['Close'].rolling(window=50).mean()
  stock_data['200_MA'] = stock_data['Close'].rolling(window=200).mean()
  ```

- Plot moving averages:

  ```python
  plt.figure(figsize=(12,6))
  ```

  plt.plot(stock_data['Close'], label='Closing Price')
  plt.plot(stock_data['50_MA'], label='50-Day MA')
  plt.plot(stock_data['200_MA'], label='200-Day MA')
  plt.legend()
  plt.title(f'{stock_symbol} Moving Averages')
  plt.show()

Step 4: Statistical Analysis & Trend Detection
- Compute daily returns:

  ```python
  stock_data['Daily Return'] = stock_data['Close'].pct_change()
  ```

- Visualize daily returns distribution:

  ```python
  sns.histplot(stock_data['Daily Return'].dropna(), bins=50, kde=True)
  plt.title('Distribution of Daily Returns')
  plt.show()
  ```

- Calculate correlation with other stocks (example: Microsoft):

  ```python
  msft = yf.download('MSFT', start='2020-01-01', end='2025-01-01')
  combined_data = pd.DataFrame()
  combined_data['AAPL'] = stock_data['Close']
  combined_data['MSFT'] = msft['Close']
  print(combined_data.corr())
  ```

Step 5: Data Visualization & Insights
- Interactive candlestick chart using Plotly:

  ```python
  import plotly.graph_objects as go
  fig = go.Figure(data=[go.Candlestick(x=stock_data.index,
                                       open=stock_data['Open'],
                                       high=stock_data['High'],
                                       low=stock_data['Low'],
                                       close=stock_data['Close'])])
  fig.update_layout(title=f'{stock_symbol} Candlestick Chart', xaxis_title='Date', yaxis_title='Price')
  fig.show()
  ```

~~~
  ```
~~~

- Box plot for price distribution:

~~~
  ```python
  sns.boxplot(y=stock_data['Close'])
  plt.title('Stock Price Distribution')
  plt.show()
  ```
~~~

5. Conclusion
This documentation provides a structured approach to retrieving, analyzing, and visualizing stock data using the Yahoo Finance API and Python. The step-by-step process ensures that any user can follow along and perform their own stock analysis efficiently.