# Real Time Stock Market Analysis

# **Process**

1.Data Collection

2.Prediction using Prophet algorithm

3.Using google cloud console prediction results integrate with power bi

4.Calculate Stock Market parameters

4.1 Risk Assessment

4.2 Risk Management

# **1.Data Collection**

**1. Overview of Financial Modeling Prep (FMP) API**

Financial Modeling Prep (FMP) provides real-time and historical financial data through a comprehensive set of APIs. It offers stock market data, financial statements, technical indicators, and economic data, making it a powerful tool for financial analysis and forecasting.

**2. API Key Integration**

To access data from FMP, an API key is required. This key authenticates requests and allows retrieval of financial data from their servers. Users must register on the FMP website to obtain an API key.

**3. Data Collected**

The following key financial data points were collected using FMP's API:

* **Stock Price Data**: Open, High, Low, Close, Volume, and Adjusted Prices.

**4. API Endpoints Used**

FMP provides various endpoints to retrieve financial data. Some of the key endpoints used in the project include:

* **Stock Price Data:**  
  **https://financialmodelingprep.com/api/v3/historical-price-full/{ticker}?apikey={API\_KEY}**

**5. Data Processing and Storage**

* Data was retrieved in JSON format and processed using **Python (Pandas, NumPy, and JSON libraries)**.
* The processed data was stored in a **Google Sheets** and used for visualization in **Power BI**.
* Additional cleaning steps included handling missing values, normalizing data, and structuring it for predictive modeling.

6.**Code**



Symbol = “Ticker”

some of examples of tickers

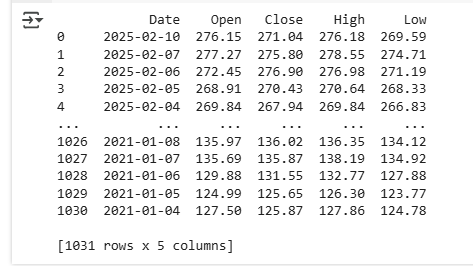
Amazon = “AMZN”

Tesla = “TSLA”

Google = “GOOGL”

JP Morgan = “JPM”

**Results**

****

# **2. Prediction Using Prophet Algorithm**

**Working Principle of the Prophet Algorithm**

Prophet is a robust time-series forecasting algorithm developed by Facebook (Meta). It is specifically designed for business forecasting and financial market trends. The algorithm effectively handles missing data, outliers, and seasonality while providing interpretable insights.

**1. Components of the Prophet Model**

Prophet follows an additive model that decomposes the time series into three main components:

y(t)=g(t)+s(t)+h(t)+ϵty(t) = g(t) + s(t) + h(t) + \epsilon\_ty(t)=g(t)+s(t)+h(t)+ϵt​

where:

* g(t)g(t)g(t) → **Trend Component** (Long-term growth patterns)
* s(t)s(t)s(t) → **Seasonality Component** (Daily, weekly, yearly cycles)
* h(t)h(t)h(t) → **Holiday Effect** (Impact of special events)
* ϵt\epsilon\_tϵt​ → **Error Term** (Unpredictable variations)

**2. Breakdown of Components**

**A. Trend Component g(t)g(t)g(t)**

* The trend component models how the data changes over time.
* Prophet uses two methods:
  1. **Linear Trend Model**: Assumes a constant growth rate.
  2. **Logistic Growth Model**: Used when there is a natural saturation or capacity limit (e.g., stock price hitting an upper bound).

Mathematical formula for **logistic growth**:

g(t)=C1+e−(k(t−m))g(t) = \frac{C}{1 + e^{- (k (t - m))}}g(t)=1+e−(k(t−m))C​

where:

* CCC = Maximum capacity
* kkk = Growth rate
* mmm = Offset parameter for inflection point

**B. Seasonality Component s(t)s(t)s(t)**

* Captures repeating patterns (daily, weekly, yearly).
* Uses **Fourier series** to approximate periodic fluctuations.

For yearly seasonality:

s(t)=∑n=1N(ancos⁡(2πnt/P)+bnsin⁡(2πnt/P))s(t) = \sum\_{n=1}^{N} \left(a\_n \cos(2\pi n t / P) + b\_n \sin(2\pi n t / P)\right)s(t)=n=1∑N​(an​cos(2πnt/P)+bn​sin(2πnt/P))

where:

* PPP = Period of seasonality (e.g., 365 days for yearly seasonality)
* an,bna\_n, b\_nan​,bn​ = Coefficients learned from data

**C. Holiday Effect h(t)h(t)h(t)**

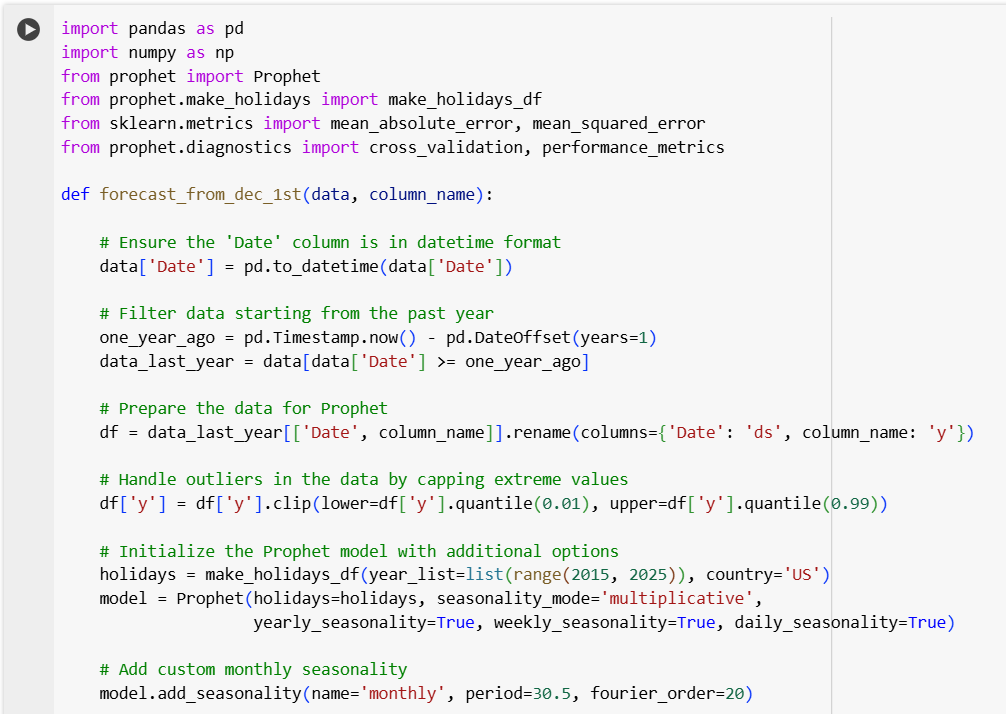
* Captures the effect of specific holidays (e.g., stock market closures).
* Prophet allows defining custom holidays that influence forecasts.

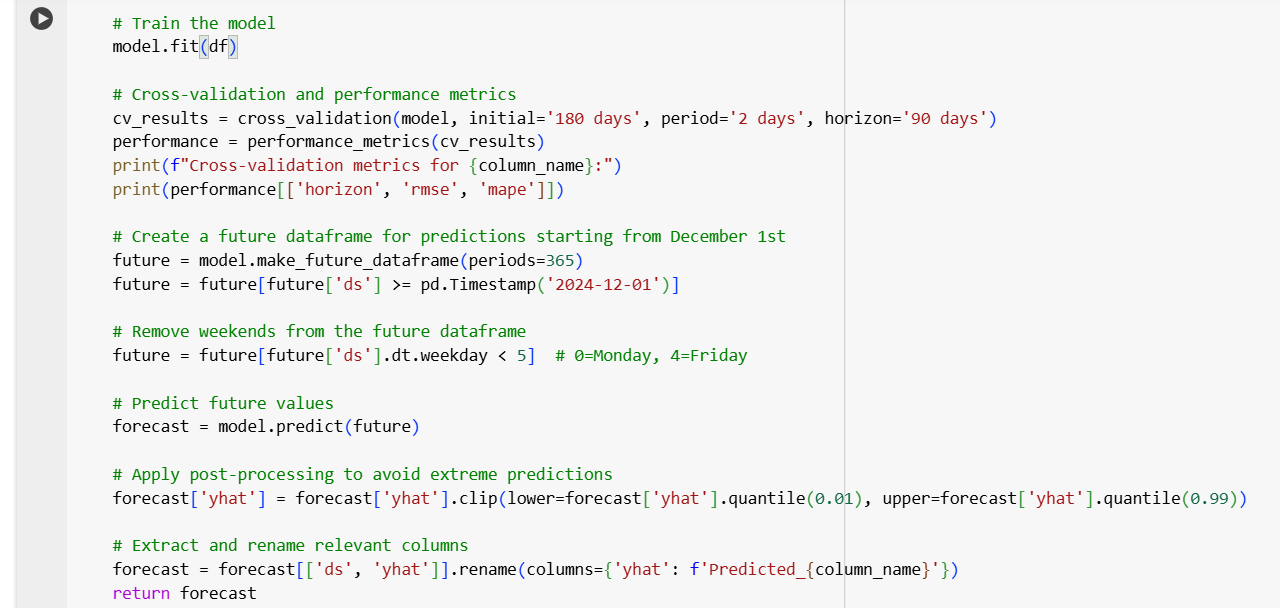
**3. Model Fitting & Forecasting**

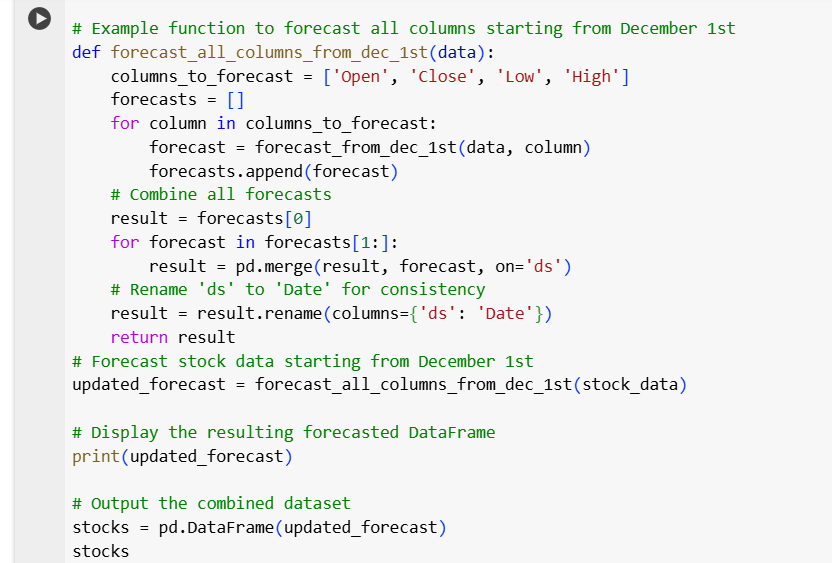
Prophet estimates the parameters of g(t)g(t)g(t), s(t)s(t)s(t), and h(t)h(t)h(t) using a Bayesian approach. The model is optimized using Stan, a probabilistic programming language.

**Steps in Forecasting:**

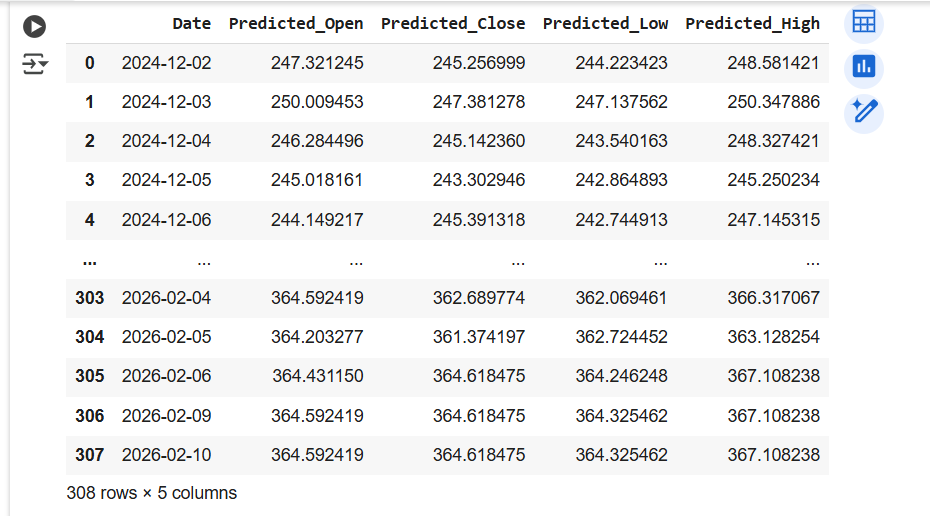
1. **Data Preprocessing**: Convert time-series data into Prophet’s format (ds → Date, y → Value).
2. **Model Training**: Fit the model with historical stock data.
3. **Future Prediction**: Generate future timestamps and predict values using estimated parameters.
4. **Uncertainty Estimation**: Provide confidence intervals around predictions.







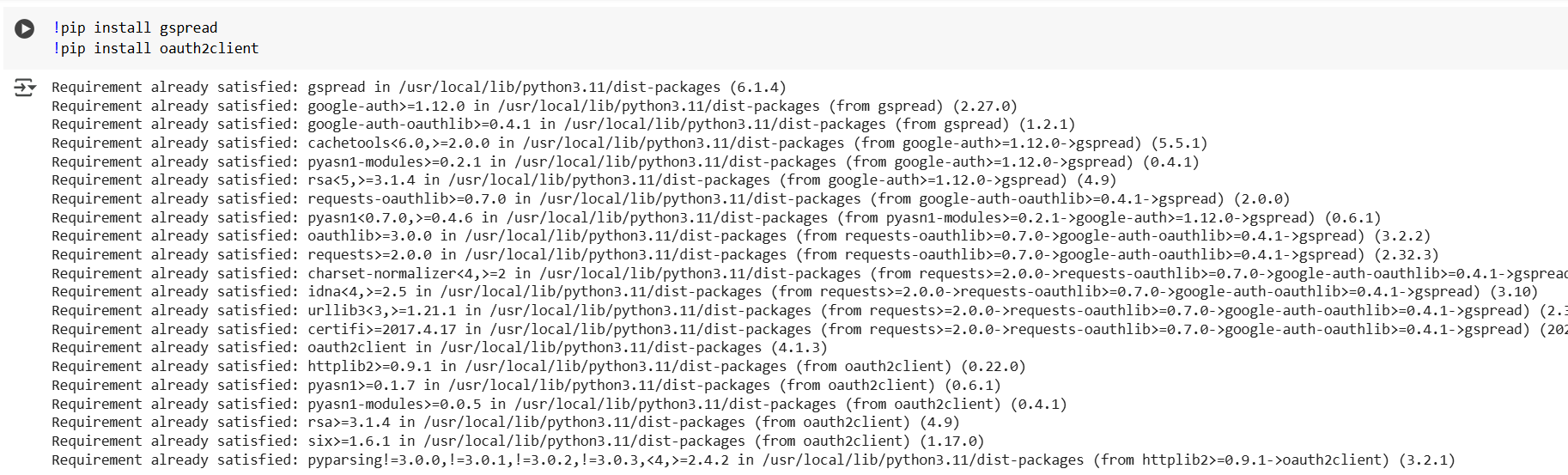
**Results**

****

# **3.Using google cloud console prediction results integrate with power bi**

**Step 1: Store Google Colab Results in Google Sheets**

**1.1 Install Required Libraries**

Google Colab does not have gspread and oauth2client installed by default. Install them using:

**1.2 Authenticate Google Drive Access**

Google Sheets API requires authentication to access your Google Sheets.

**1.3 Create or Open a Google Sheet**

If you already have a Google Sheet, you can open it using its name. Otherwise, create a new one.

**1.4 Save DataFrame to Google Sheets**

After performing calculations in Colab (such as stock market analysis), store results in Google Sheets.



**Step 2: Enable Google Sheets API in Google Cloud Console**

To connect Google Sheets with Power BI, enable Google Sheets API on Google Cloud Console.

**2.1 Enable Google Sheets API**

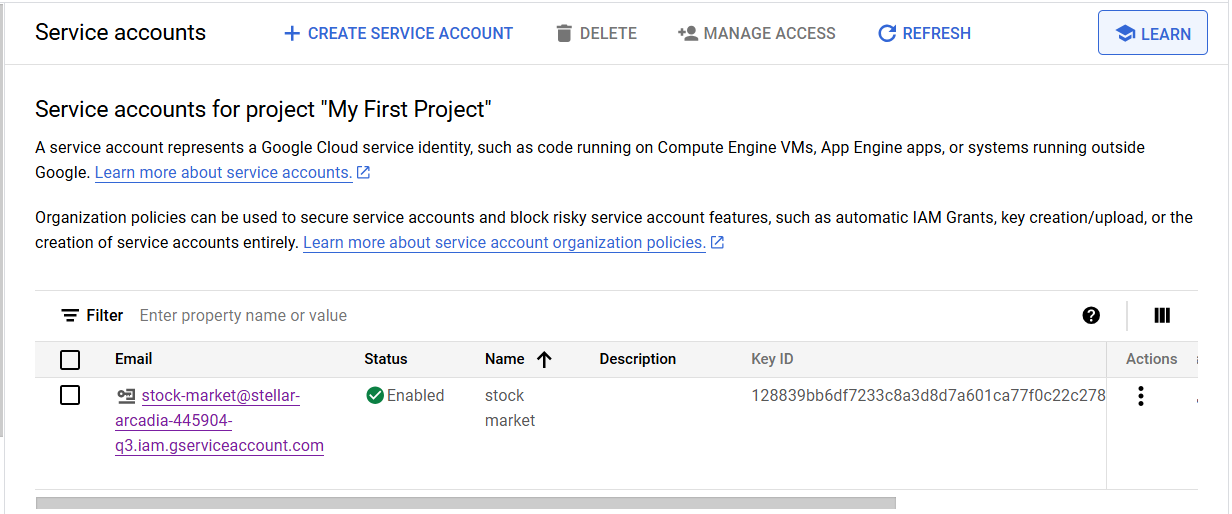
Go to Google Cloud Console.

Create a new project or select an existing one.

Navigate to APIs & Services → Library.

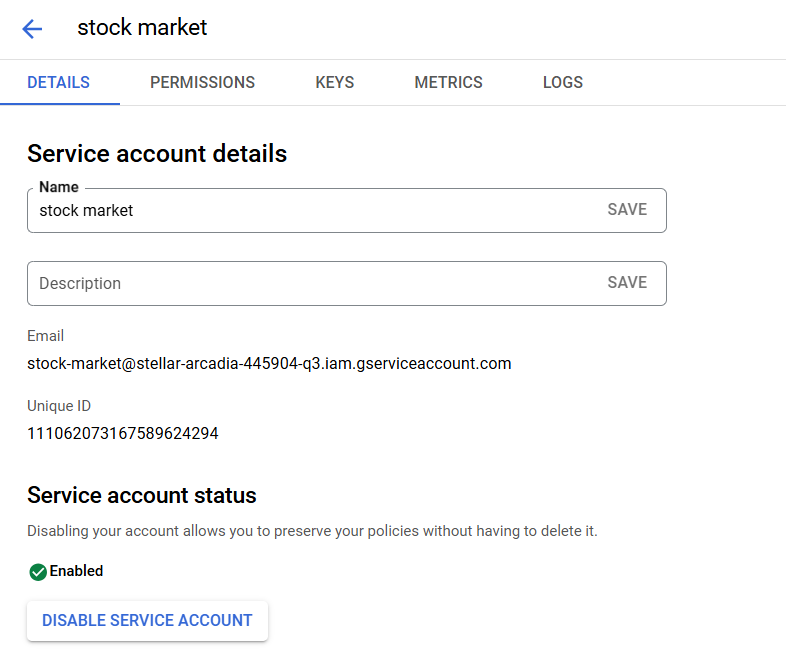
Search for Google Sheets API and enable it.

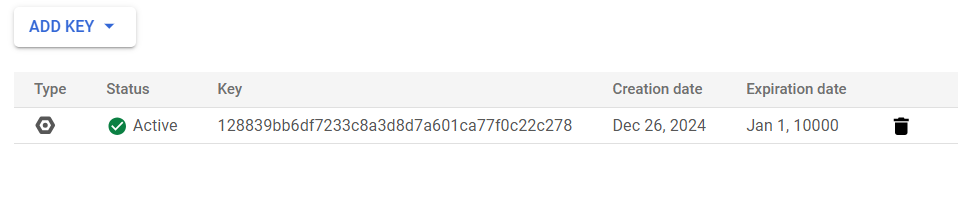
Similarly, enable Google Drive API (required for Power BI integration).

  
**2.2 Create a Service Account**

1. Go to **APIs & Services** → **Credentials**.
2. Click **Create Credentials** → **Service Account**.
3. Provide a name and description → Click **Create & Continue**.
4. Assign **Editor** role → Click **Done**.
5. Open the **Service Account** and go to **Keys**.
6. Click **Add Key** → **Create new key** → Select **JSON** → **Download**.

📂 **Save this JSON file securely**. It will be used for authentication.





**Step 3: Integrate Google Sheets with Power BI**

Once Google Sheets API is set up, connect it with Power BI.

**3.1 Open Power BI Desktop**

1. Launch **Power BI Desktop**.
2. Click **Get Data** → **More**.
3. Search for **Google Sheets** or **Web**.

**3.2 Connect Using Google Sheets URL**

1. Open the **Google Sheet** storing stock data.
2. Click **File** → **Share** → **Get link**.
3. Change access to **Anyone with the link** → Copy the link.
4. Extract the **Google Sheet ID** from the link:

https://docs.google.com/spreadsheets/d/XXXXXXXXXXXXXXXXXXXXXXXX/edit

* + The **Sheet ID** is the part after /d/ and before /edit.

1. Use this URL in Power BI:

https://docs.google.com/spreadsheets/d/XXXXXXXXXXXXXXXXXXXXXXXX/gviz/tq?tqx=out:csv

**3.3 Load Data in Power BI**

1. In Power BI, select **Get Data** → **Web**.
2. Paste the **modified Google Sheets URL**.
3. Click **OK** → Load data.
4. Now, Power BI will pull data directly from **Google Sheets**.

**Step 4: Automate Data Updates**

To keep the Power BI dashboard updated:

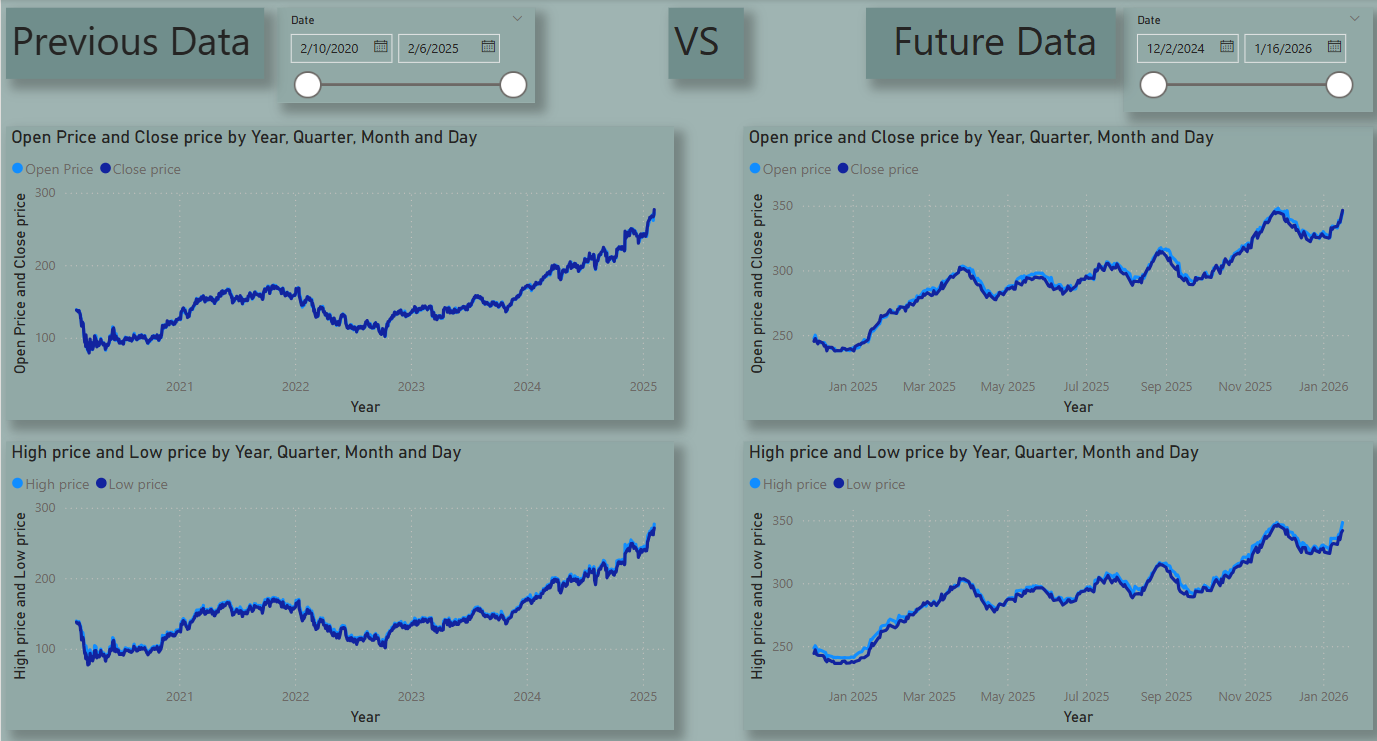
* **Google Colab** automatically pushes new stock data to **Google Sheets**.
* **Power BI** refreshes from **Google Sheets** at scheduled intervals.

To enable auto-refresh in Power BI:

1. Go to **Power BI Service**.
2. Open **Dataset Settings**.
3. Enable **Scheduled Refresh** every 1 hour (or desired frequency).

# **4.Calculate Stock Market parameters**

**OverView**

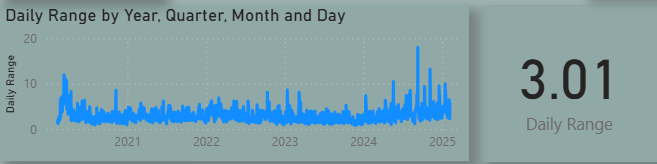
****

**4.1 Risk Assessment**

**4.1.1Daily Range**

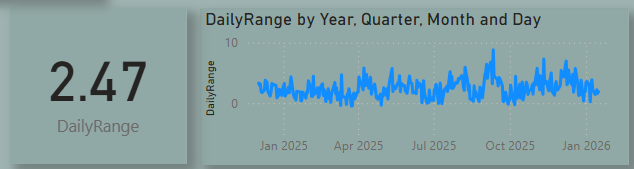
**Past Data**

****

****

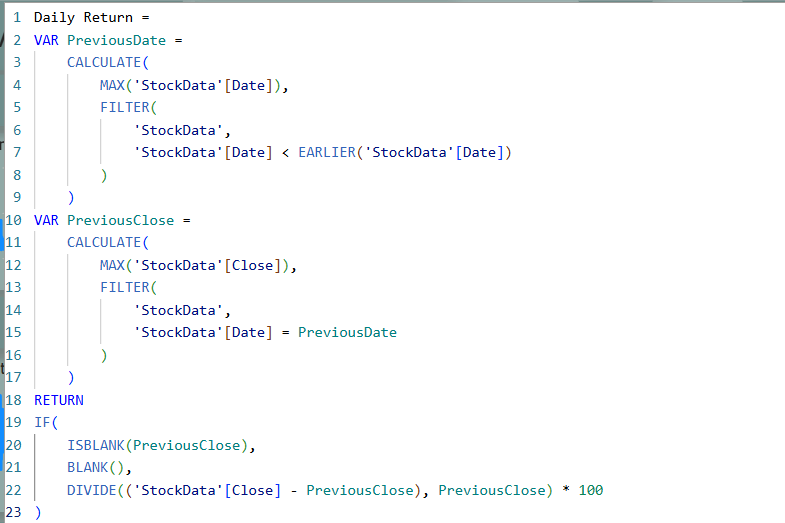
**Future Data**

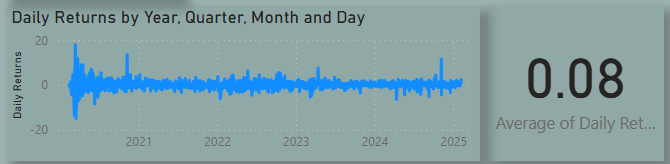
****

****

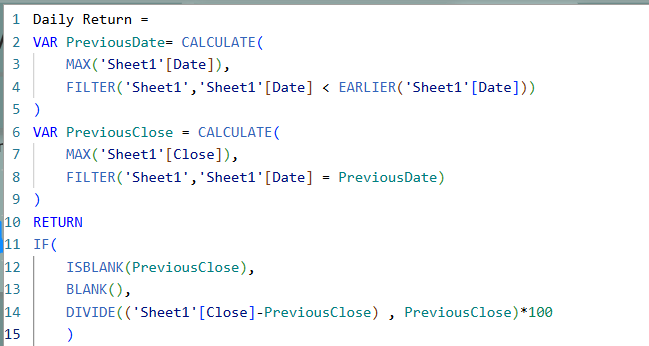
**4.1.2 Daily Returns**

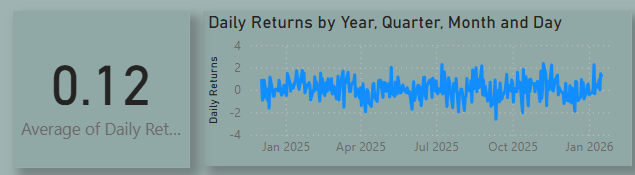
**Past Data**

****

****

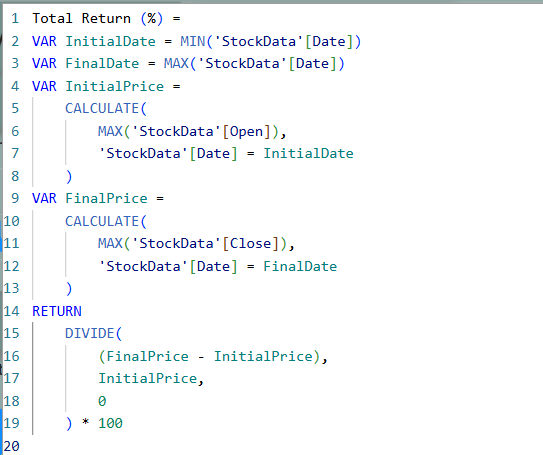
**Future Data**

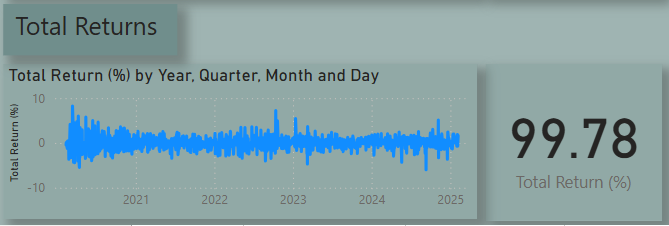
****

****

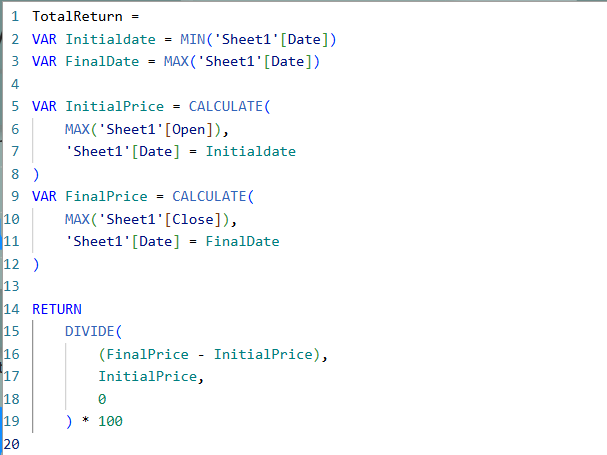
**4.1.3 Total Returns**

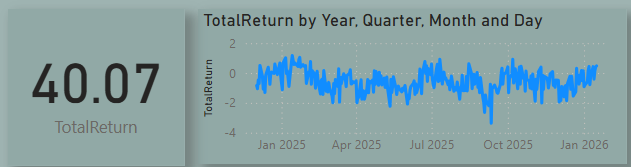
**Past Data**

****

****

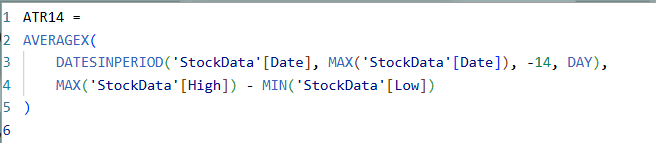
**Future Data**

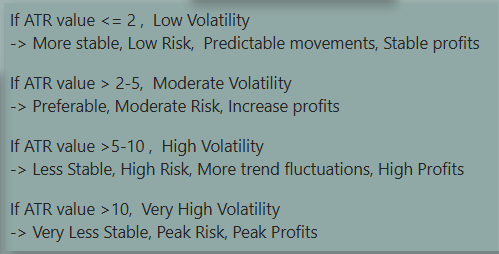
****

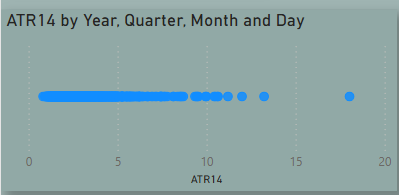
****

**4.1.4 Average True Range**

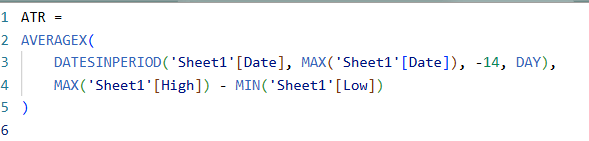
**Past Data**

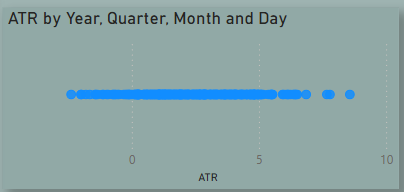
****

****

****

**Future Data**

****

****

**4.1.5 Risk Reward Ratio**

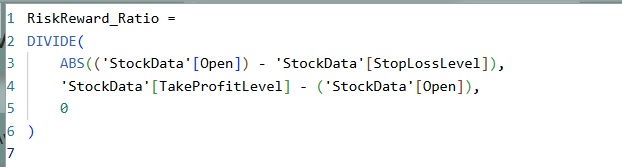
**Past Data**

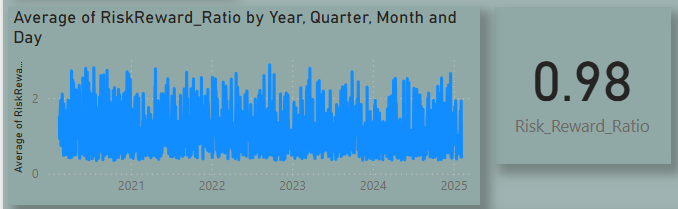
**4.1.5.1 Stop Loss Level**

****

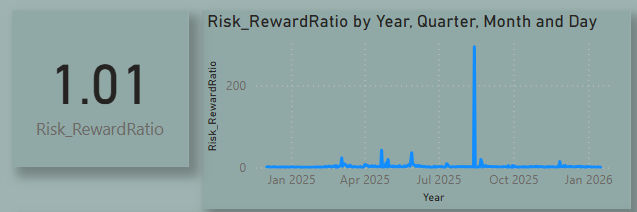
**4.1.5.2 Take Profit Level**

****

****

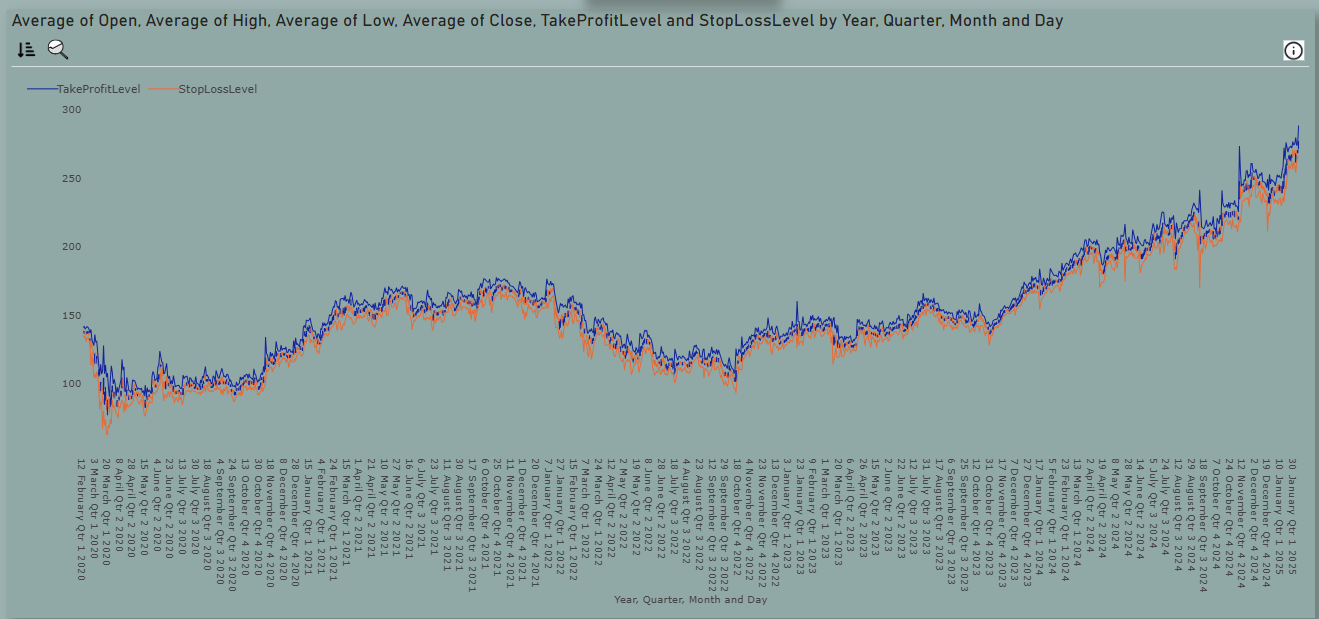
****

**Future Data**

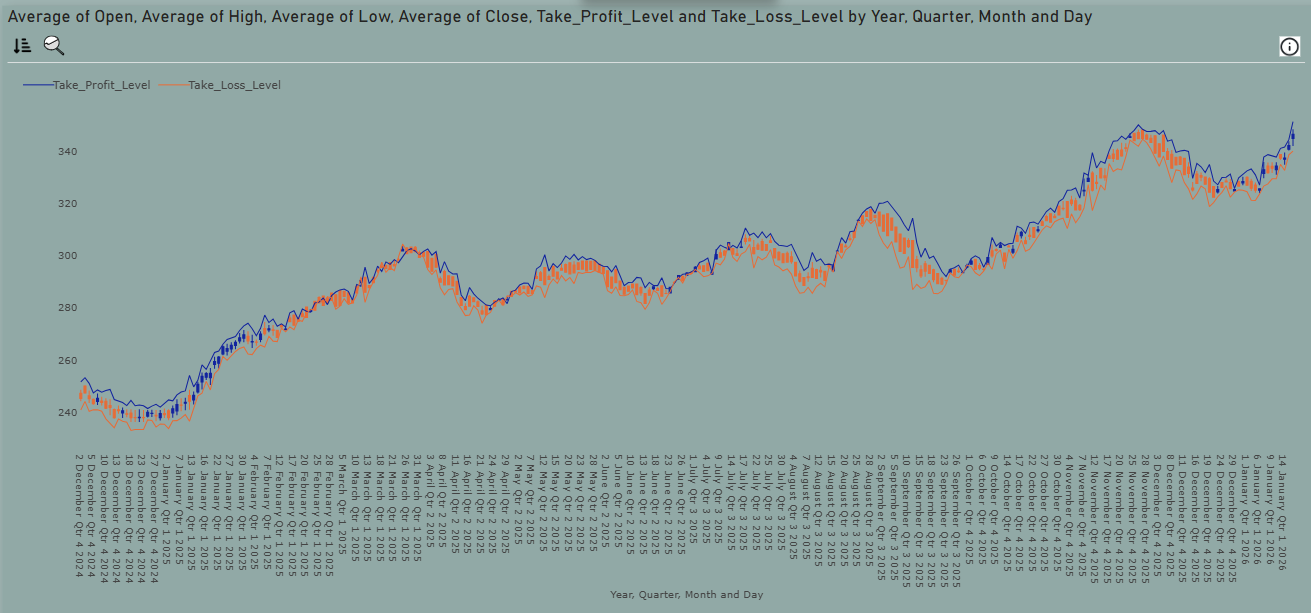
****

**4.1.6 Stop Loss Level and Take Profit Level**

**Past Data**

****

**Future Data**

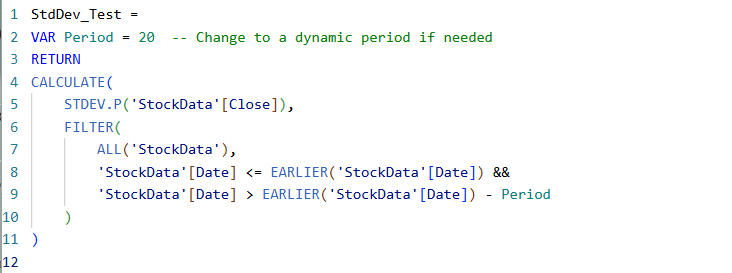
****

**4.2 Risk Management**

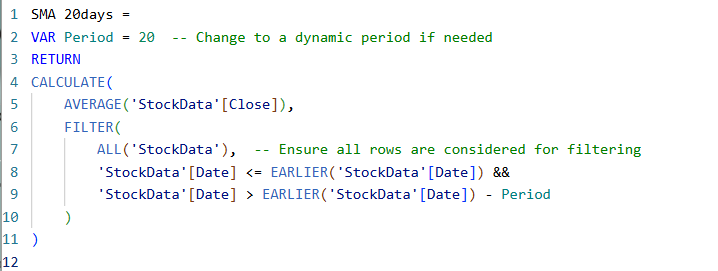
**4.2.1 Bollinger Bands**

**Past Data**

**4.2.1.1 Standard Deviation**

****

* + - 1. **Simple Moving Average**

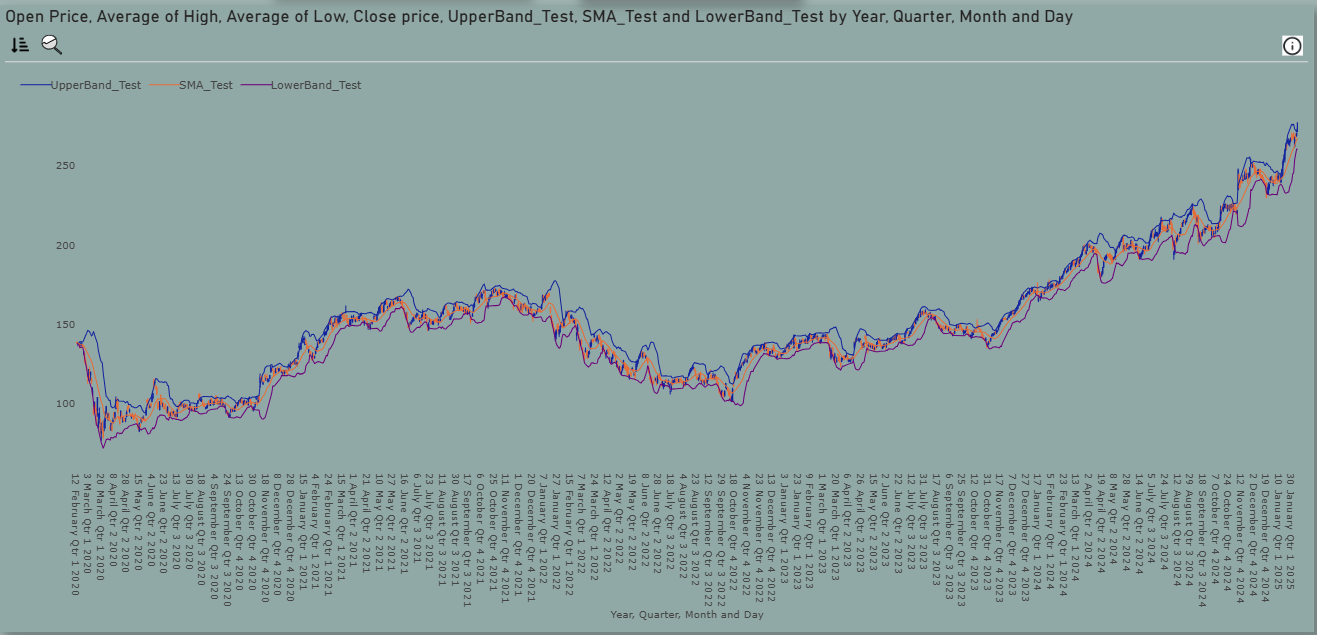
****

* + - 1. **Upper Band Test**

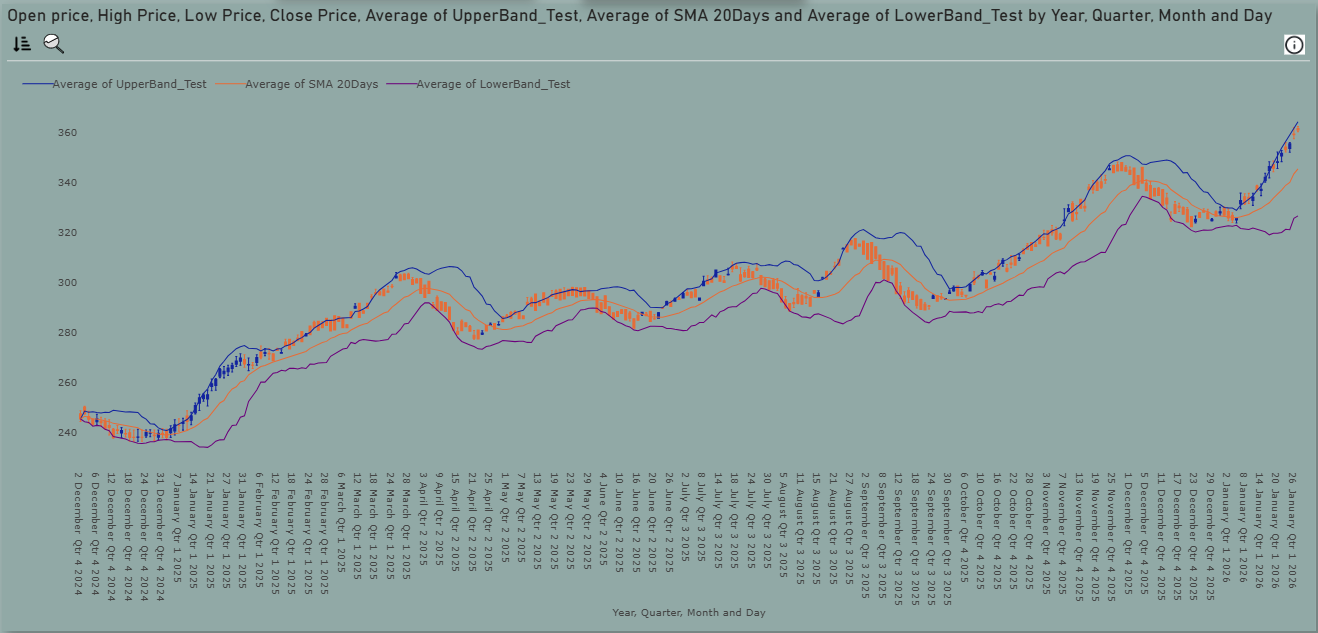
****

* + - 1. **Lower Band Test**

****

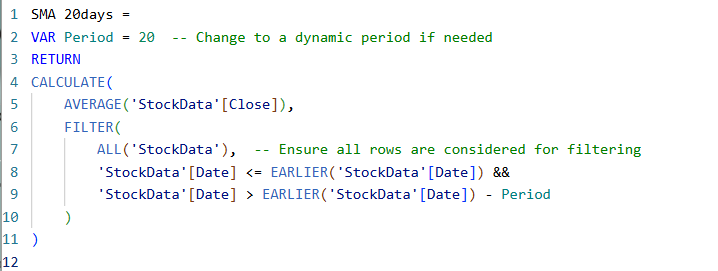
****

**Future Data**

****

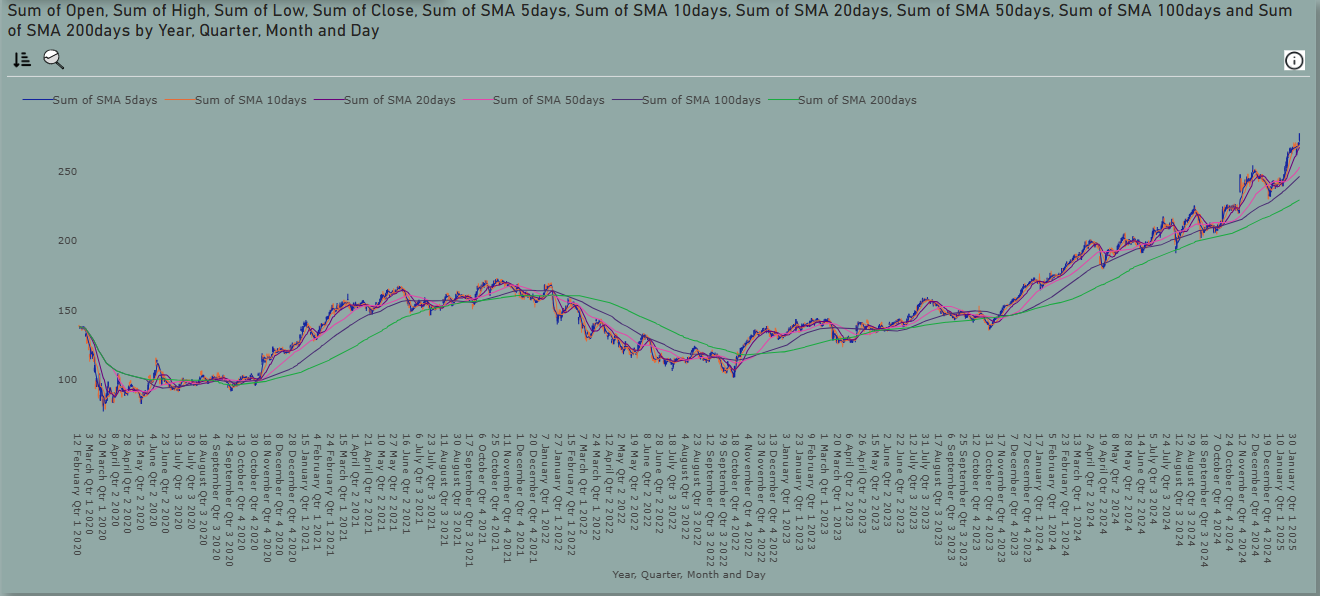
* + 1. **Simple Moving Average**

**Past Data**

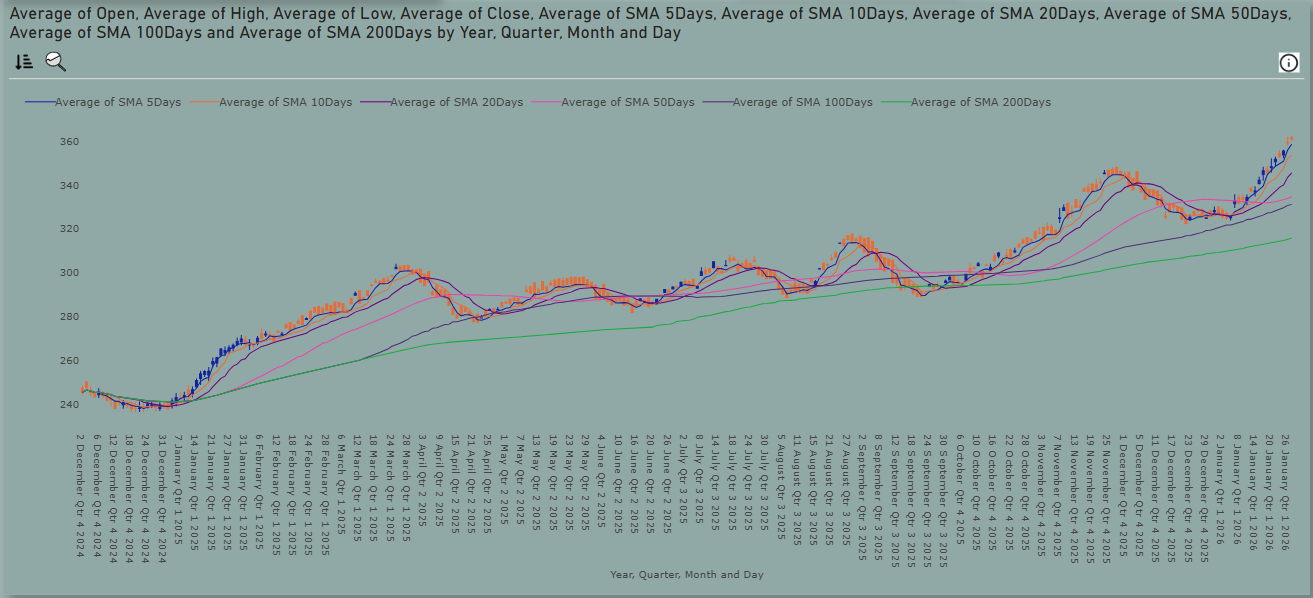
****

**5, 10, 20, 50, 100, 200 Days Simple Moving Averages**

****

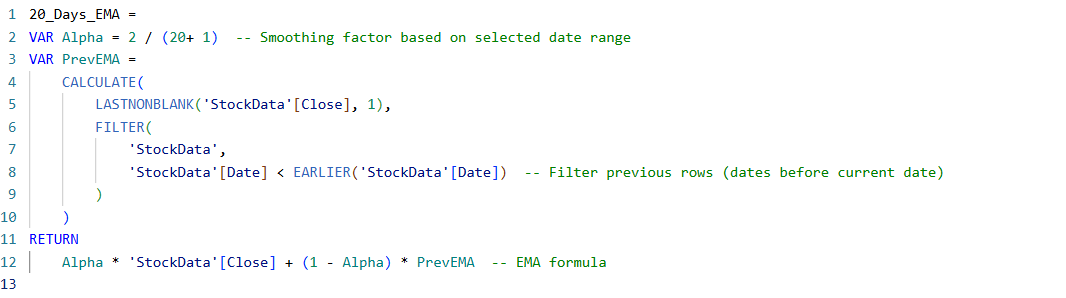
****

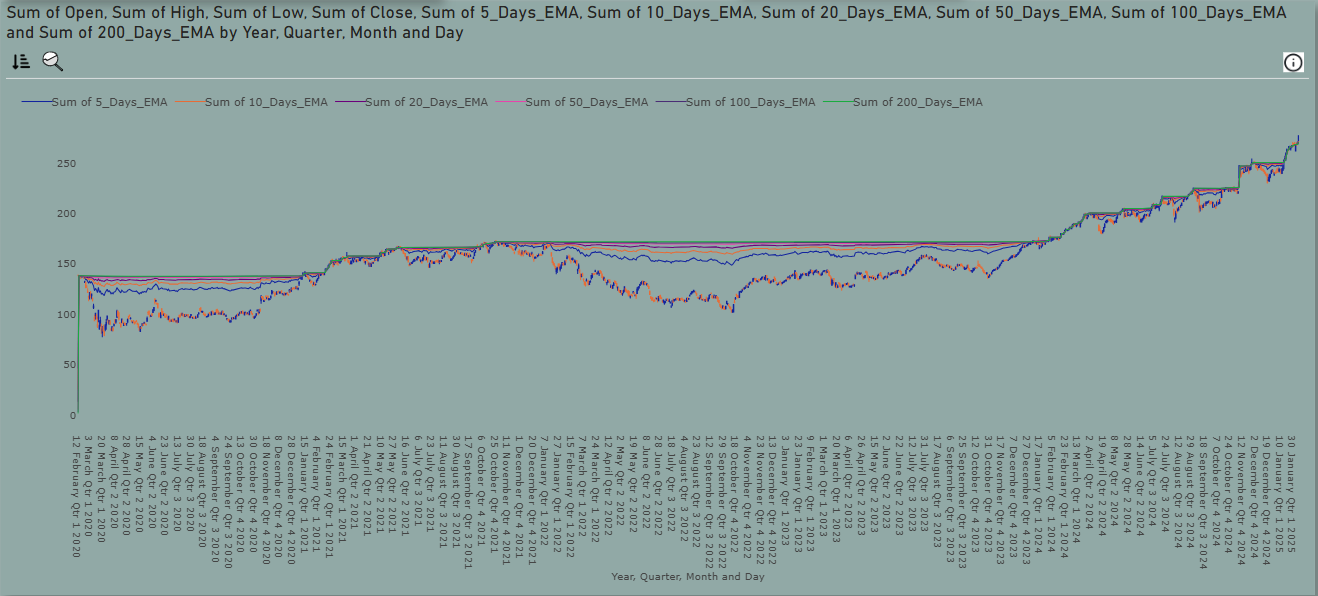
**Future Data**

****

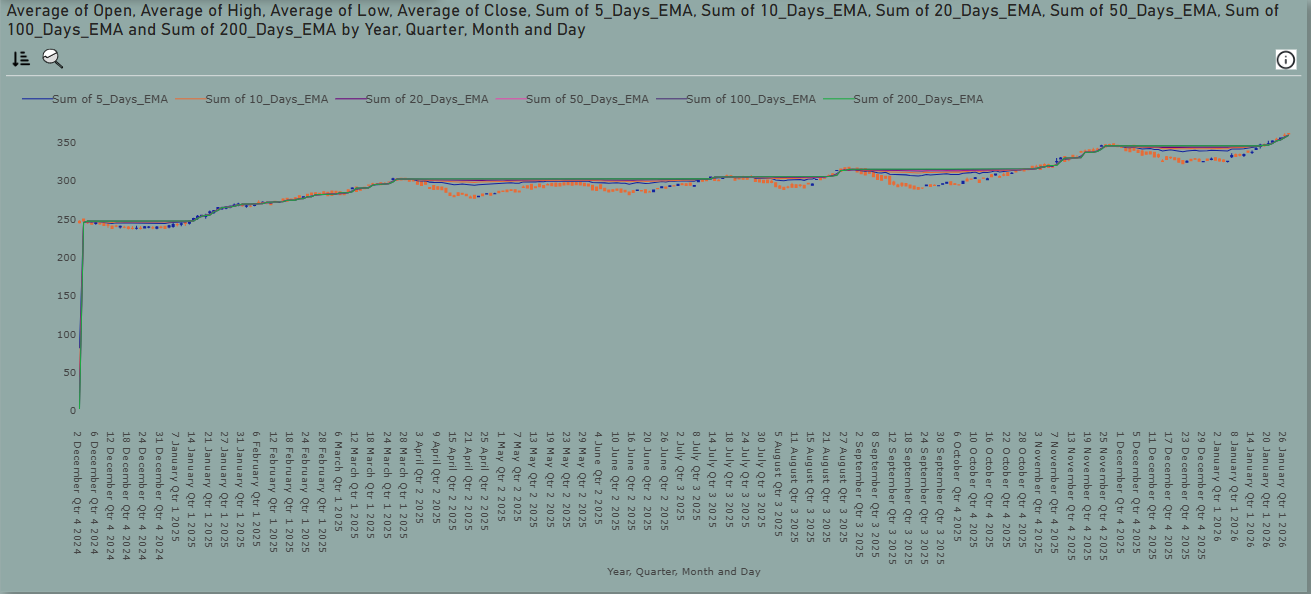
* + 1. **Exponential Moving Average**

**Past Data**

****

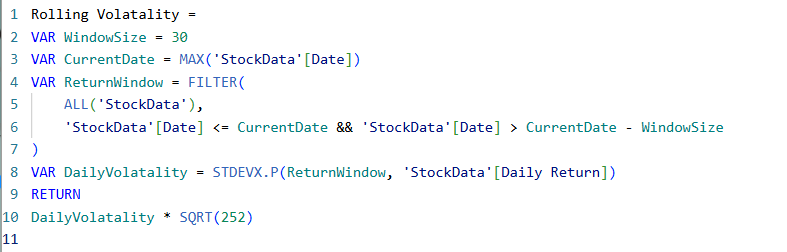
****

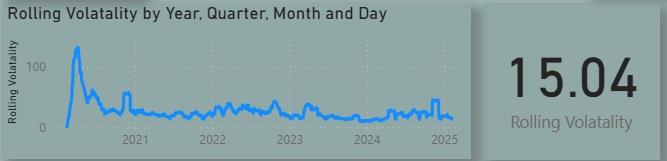
**Future Data**

****

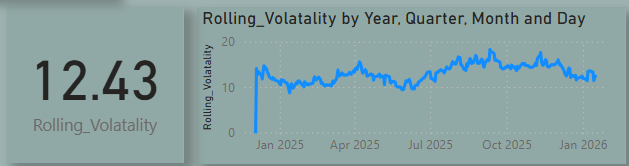
* + 1. **Volatility**

**Past Data**

****

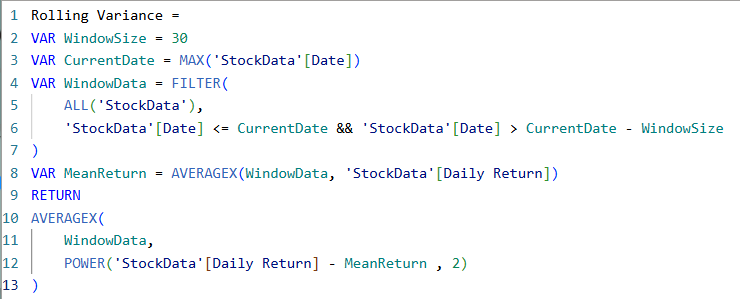
****

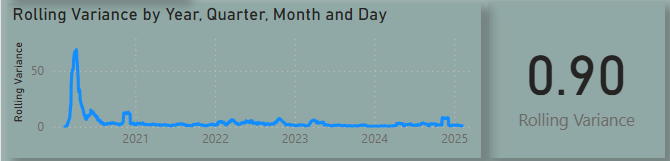
**Future Data**

****

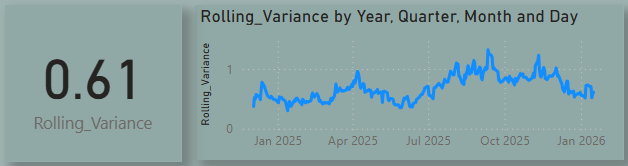
* + 1. **Variance**

**Past Data**

****

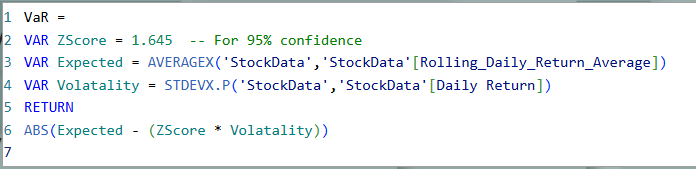
****

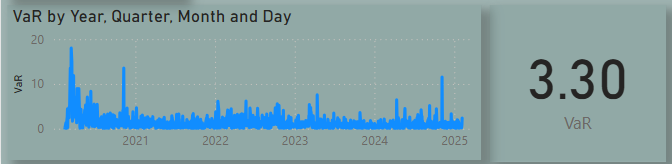
**Future Data**

****

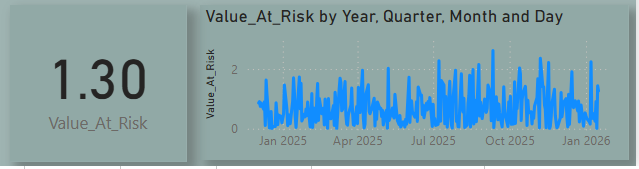
* + 1. **Value at Risk**

**Past Data**

****

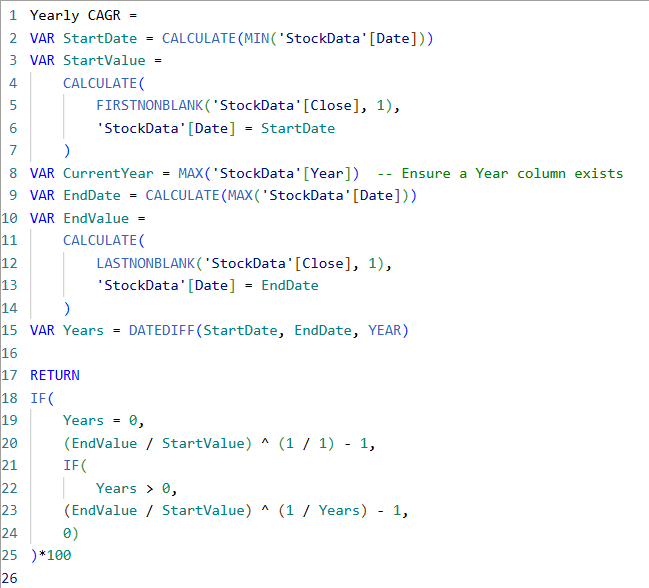
****

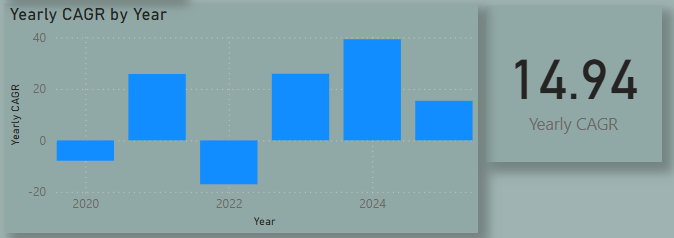
**Future Data**

****

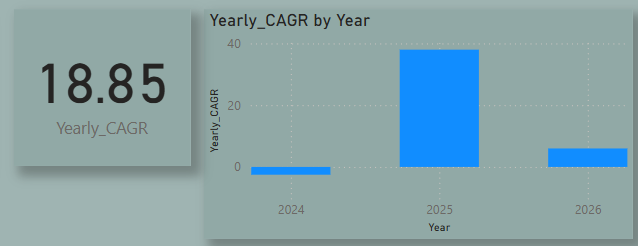
* + 1. **CAGR**

**Past Data**

****

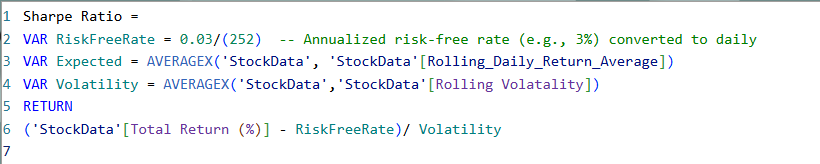
****

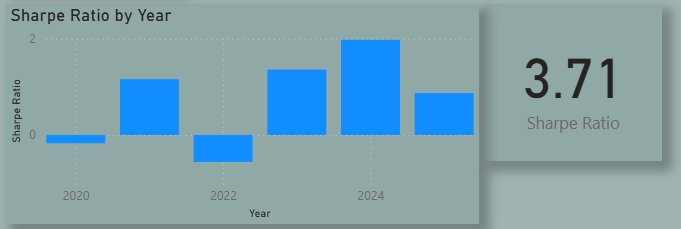
**Future Data**

****

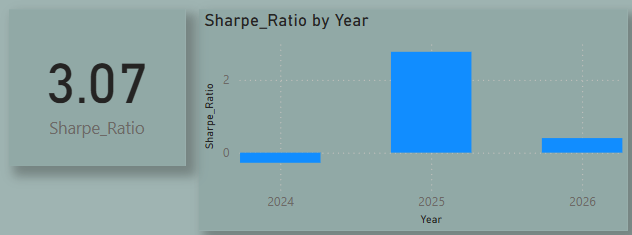
**4.2.8 Sharpe Ratio**

**Past Data**

****

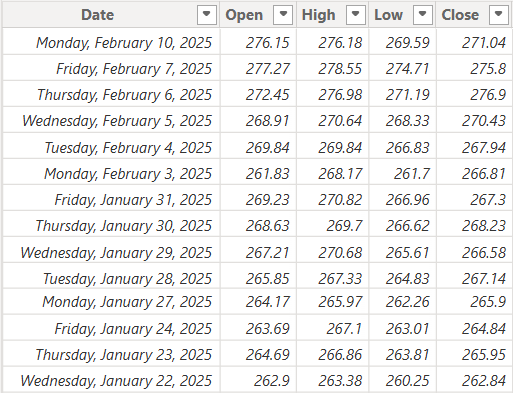
****

**Future Data**

****

# **Sample actual data vs Predicted Data Prices**

# Past Data



# Future Data

