Project GitHub Link: https://github.com/rickygarim/si-206-final-project

In addition to your API activity results, you will be creating a report for your overall project. The report must include:

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

Aya: The primary goal of this project was to analyze stock market news sentiment for specific companies like Apple (AAPL), Microsoft (MSFT), and Tesla (TSLA). To achieve this, the MarketAux API was used to gather relevant news articles about these companies. The sentiment of each article was then analyzed using the VADER SentimentIntensityAnalyzer from the nltk library to determine if the tone of the article was positive, negative, or neutral. The collected data, including sentiment scores, was saved into a database to ensure it could be stored and accessed for future use. Finally, visualizations were created to show how sentiment scores changed over time for each stock ticker, helping to identify trends and patterns in the news sentiment.

Venkat: The main goal was to find and compare metrics that are related to SP500 and could impact the overall index and investor . The data we planned to collect were DIX and GEX metrics. We initially planned to work with the following website to retrieve these metrics: https://squeezemetrics.com/monitor/dix.

Ricky: I integrated an API to retrieve stock price data, which was then stored in a database table for further processing. Using the stored data, I calculated a moving average over a specified window and updated the table with these calculated values. Finally, I visualized both the original stock prices and their corresponding moving averages on a chart, allowing for a clear comparison of the data trends over time.

2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

Aya: Stock news articles were fetched month by month for eight stock tickers (AAPL, MSFT, TSLA, GOOGL, AMZN, NFLX, ORCL, and ADBE) using MarketAux API. Sentiment analysis were preformed on the collected articles user VADER SentimentIntensityAnalyzer to determine the sentiment score. The results, including the ticker, article title, description, timestamp, and sentiment scores were stored in an SQLite database. Finally, visulizations were created using Matplotlib to show how sentiment scores changed over time for each stock ticker in the date range (Sep 2024 to Dec 2024).

Venkat: We achieved what we initially set out to do, but had to make a change in the metrics used. We eventually set on VIX (to measure volatility) and TNX (to measure 10-year Treasury bond interest rates). Both of these metrics could be used to measure investor sentiment and compare and contrast how that impacts each other and how they impact the overall performance of the index, allowing us to achieve what we initially set out to do.

Ricky: The goals of the project were successfully achieved by integrating a stock price API to gather real-time stock data, which was stored in a SQLite database. I used the Alpha Vantage API to fetch daily stock prices, including the closing prices and corresponding dates. After storing the data, I calculated a moving average over a defined window (e.g., 3 days) for each stock price. Both the raw stock prices and the calculated moving averages were then visualized on a chart to show trends and patterns. This project involved data gathering, processing, and visualization, providing insights into stock performance over time.

3. The problems that you faced (10 points)

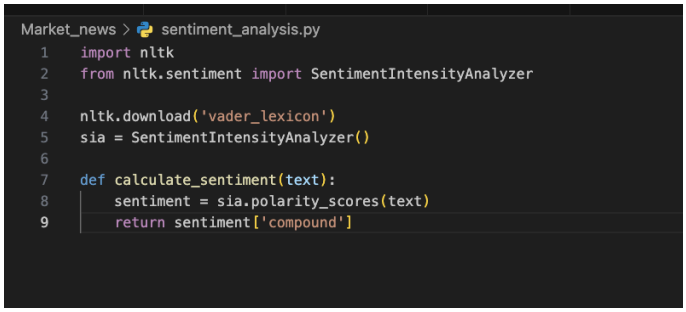
Aya: One challenge was API rate limiting, MarketAux API restricts the number of requests allowed per day for news. To solve this problem, the number of articles fetched per month was limited to three per stock ticker. Another issue was duplicate entries, which occurred when articles overlapped in date ranges. This was fixed by adding a UNIQUE constraint in the database to prevent duplicates.

Venkat: The initial website (https://squeezemetrics.com/monitor/dix) I set out to gather data from was very difficult to work with. There was no API to call to retrieve the needed data and rather everything was stored in a single CSV file that could be downloaded. However, it was not easy to use BeautifulSoup with this website as the data to download is dynamically generated each time the download button is clicked which can't be directly read by BeautifulSoup. There had to switch the website - and subsequently data - we set out to collect (this website was one of the only website for the two initially selected metrics as they as very specific not general metrics).

Ricky: The Alpha Vantage API has rate limits, which meant that I had to manage requests carefully to avoid hitting these limits. Making a new account wasn't possible since after rate limiting, they seem to block your Ip address from making additional requests. I had to use a vpn to finish testing.

4. The calculations from the data in the database (i.e. a screen shot) (10 points)
Aya:

id	ticker	title
1	AAPL	Top 13 Expat Employment Industries in the US
2	AAPL	Apple Inc. (AAPL) Q4 2024 Earnings Call Transcript
3	AAPL	Top 10 AI Stocks on Investors' Radar These Days
4	AAPL	Kartoon Studios Announces $6.5 Million Offering By Investing.com
5	MSFT	Top 13 Expat Employment Industries in the US
6	MSFT	BlackRock launches ETF that expands beyond the 'Magnificent Seven'
7	MSFT	2 Top Artificial Intelligence (AI) Stocks to Buy Right Now
8	MSFT	CS.ai director John Dykes sells $191,874.1K in stock By Investing.com
9	TSLA	Lucid Group: Continued Liquidity Dip Sets Up Precarious Drive For This EV Upstart (LCID)
10	TSLA	Judge aims to rule on Elon Musk's $56 billion Tesla pay by year end By Reuters
11	TSLA	JPMorgan Jumps Last against Tesla ever Musk tweet, warrants
12	TSLA	US Stocks Close Higher With Fed Rate-Cut Expected: Markets Wrap
13	GOOGL	Top 13 Expat Employment Industries in the US
14	GOOGL	OpenAI brings search features to ChatGPT in challenge to Google
15	GOOGL	Alphabet Inc. (GOOG)'s DeepMind Develops AI System for Real-Time Digital Personality Replication
16	GOOGL	Nasdaq closes at record as investors prepare for Fed rate decision
17	AMZN	Andreas Rafaeloan's Firm Expands Grip in BioMarin Pharmaceutical Inc Holding
18	AMZN	Amazon reports $158B in Q3 revenue, beats Wall Street expectations
19	AMZN	Amazon.com, Inc. (AMZN) Leverages Generative AI to Enhance Same-Day Delivery Efficiency
20	AMZN	US stock futures drift lower with Fed rate decision in focus By Investing.com
21	NFLX	Netflix (NFLX) Gains But Lags Market: What You Should Know
22	NFLX	Dow Jones Futures: Apple, Amazon Lead Earnings Movers After Stock Market Sell-Off; Jobs Report Looms
23	NFLX	Up 500% in the Past Year: Why Investors Are Falling in Love With This Hypergrowth SpaceX Competitor
24	NFLX	Kartoon Studios Announces $6.5 Million Offering By Investing.com
25	AAPL	Top Research Reports for Apple, Tesla & T-Mobile US
26	AAPL	Finding Long-Term Winners in Packaging and Exchange Stocks
27	AAPL	Up 500% in the Past Year: Why Investors Are Falling in Love With This Hypergrowth SpaceX Competitor
28	AAPL	Bitcoin, Ethereum, Dogecoin Settle Down Amid Growing Talks Of A Strategic BTC Reserve: Top Analyst Sees Prospect Of Apex Crypto Hitting $150K This Week -
29	MSFT	10 Fastest Countries to Emigrate to from India



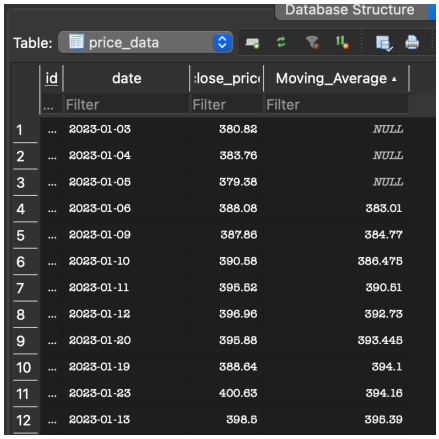
```
Market_news > sentiment_analysis.py
1  import nltk
2  from nltk.sentiment import SentimentIntensityAnalyzer
3
4  nltk.download('vader_lexicon')
5  sia = SentimentIntensityAnalyzer()
6
7  def calculate_sentiment(text):
8      sentiment = sia.polarity_scores(text)
9      return sentiment['compound']
```

Venkat:


```
{
  "VIX": {
    "Average Value": 19.56,
    "Minimum Value": 15.78,
    "Maximum Value": 26.52,
    "Number of Entries": 100
  },
  "TNX": {
    "Average Value": 3.6,
    "Minimum Value": 3.29,
    "Maximum Value": 4.07,
    "Number of Entries": 100
  },
  "Summary": {
    "Analysis Period": "2023-01-03 to 2023-05-25",
    "Total Days Analyzed": 100
  }
}
```


JSON created from calculations from data in the database of the two metrics (VIX and TNX). This JSON file is also included in the Git/Canvas files.

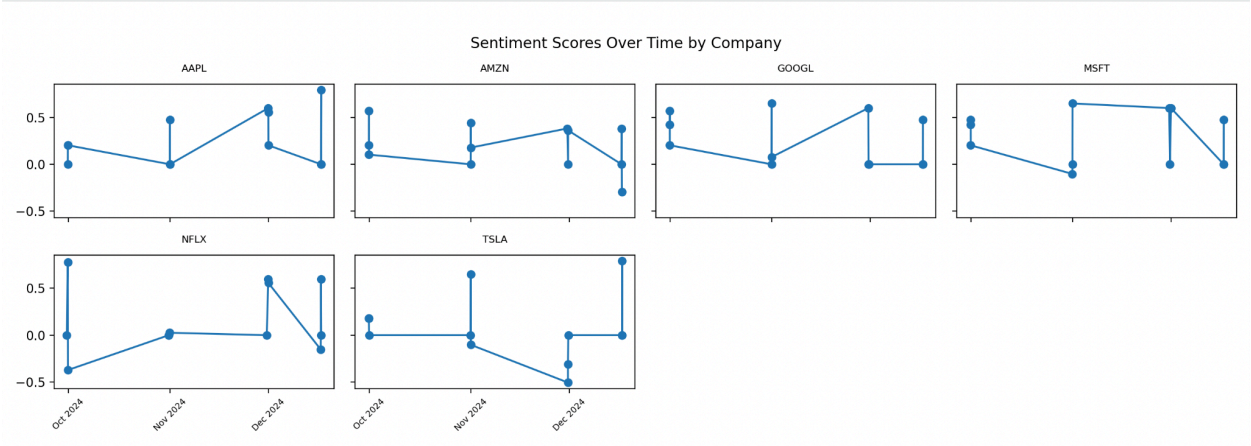
Ricky:



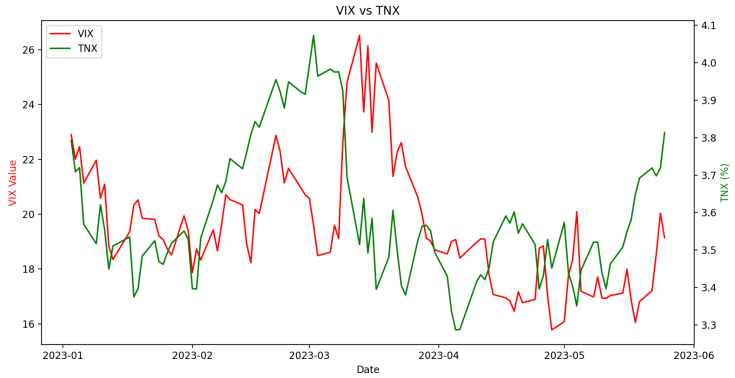
	id	date	lose_pric	Moving_Average
	...	Filter	Filter	Filter
1	...	2023-01-03	380.82	NULL
2	...	2023-01-04	383.76	NULL
3	...	2023-01-05	379.38	NULL
4	...	2023-01-06	388.08	383.01
5	...	2023-01-09	387.86	384.77
6	...	2023-01-10	390.58	386.475
7	...	2023-01-11	395.82	390.51
8	...	2023-01-12	396.96	392.73
9	...	2023-01-20	395.88	393.445
10	...	2023-01-19	388.64	394.1
11	...	2023-01-23	400.63	394.16
12	...	2023-01-13	398.5	395.39

Moving average calculation at a specific point in time using the last 3 days for averaging.

5. The visualization that you created (i.e. screen shot or image file) (10 points)
Aya:

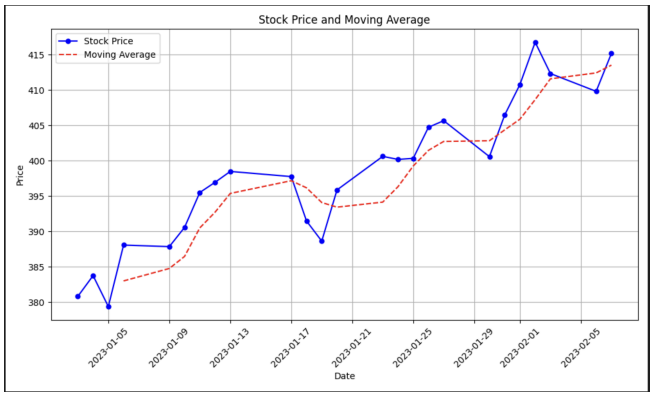


Venkat:



Graph created by plotting the VIX score (red) and TNX percentage (green) for the exact same days across the same time span (first 100 market days of 2023). Graph included in Git/Canvas

Ricky:



Plots the moving average along with SPY's price.

6. Instructions for running your code (10 points)

Aya:

1. **Install the required libraries**:
   Run the following command in your terminal:
   `pip install requests beautifulsoup4 matplotlib nltk`
2. **Run the main script multiple times** to fetch and save news data:
   `python main.py`
3. **Generate visualizations**:
   Use the following command to run the visualization script:
   `python visualize.py`

Venkat: Need to have pip installed the following libraries: python, matplotlib, sqlite3, datatime, yfinance (main wrapper library that is used to make API called to retrieve the VIX and TNX metrics). Following this, there are two different files to each retrieve either the VIX data or the TNX data and stored in two different tables in the same database (these can be run in any order). The two files will create a table if there is not one and will fetch 25 entries each time it is ran. There is a third visualization file that will read from the created database and create the graph and the JSON shown above (of course, this can only be run after running the two data fetching files).

Ricky: Need to have pip installed the following libraries: requests, sqlite3, datetime, pandas, os, and matplotlib.pyplot. To run, type in ```python3 Market-Data/main.py``` in the terminal. Every time you run this command, the api is called which fetches another 25 items of data, does the necessary calculations, and also visualizes the updated data.

7. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)
Aya: The fetch_stock_news_monthly function retrieves and filters articles, while the clean_text function cleans any HTML content. Sentiment analysis is performed using the calculate_sentiment function, and results are stored in the database with save_news_to_db. Finally, the visualize_sentiment_subplots function generates a dynamic set of subplots using Matplotlib to show sentiment scores over time for each stock ticker. The script is easy to run with minimal setup and produces a clear visualization of sentiment trends for analysis.

1. Fetch_stock_news_monthly: Fetches stock news articles for a given stock ticker, month-by-month, within a specified date range.

2. Clean_text: Cleans a given HTML string by removing HTML tags and returning plain text.

3. calculate_sentiment: Calculates the sentiment score of a given text using the VADER SentimentIntensityAnalyzer.

4. save_news_to_db: Saves news articles and their sentiment scores into an SQLite database, avoiding duplicates.

5. visualize_sentiment_subplots: Generates and displays subplots of sentiment scores over time for each stock ticker.

Venkat: No specific functions were created for retrieving or storing the VIX or TNX data. However, the general workflow of both files is as follows: define the needed constants first, connect to the DB, create table if none, check last entry in data table, process the next range of needed entries, fetch next range of neede entries from database, format as needed, push to datatable, log/print as needed to show what occurred. The visualization file works similarly: define constants initially, connect to datatable, read in entries, create plots, create JSON.

Ricky:

ApiPipeline Class

1. **__init__(self, market_data_key, db_name)**
   Initializes the ApiPipeline class with the market data API key and the database name.
2. **_get_latest_date(self)**
   Fetches the most recent stored date from the database. Returns the latest date as a string or None if no data exists.
3. **_get_stock_data(self)**
   Fetches stock data from the Alpha Vantage API. Returns a dictionary of the stock's time series data.
4. **fetch_data(self)**
   Fetches stock data, starting from the latest available date in the database, and returns the next 25 records with the date and close price.

DataProcessingPipeline Class

1. **__init__(self, window_size, db_path)**
   Initializes the DataProcessingPipeline class with the window size for calculating the moving average and the database path.
2. **_fetch_data(self)**
   Fetches stock data from the database and returns it as a pandas DataFrame.
3. **_calculate_moving_average(self, df)**
   Calculates the moving average based on the provided window size and adds it as a new column to the DataFrame. Returns the DataFrame with the new moving average column.
4. **process_data(self)**
   Fetches the raw data, processes it by calculating the moving average, and saves the updated data back into the database.

DatabasePipeline Class

1. **_setup_database(self)**
   Sets up the database by creating a table for storing stock data if it does not already exist.

2. **__init__(self, db_name)**
   Initializes the DatabasePipeline class with the database name and sets up the database.
3. **save_data(self, data)**
   Inserts new stock data into the database, ignoring any records with existing dates.

VisualizationPipeline Class

1. **__init__(self, db_path)**
   Initializes the VisualizationPipeline class with the path to the database.
2. **_fetch_data_from_db(self)**
   Fetches stock data from the database, including the close price and moving average, and returns it as a pandas DataFrame.
3. **visualize_data(self)**
   Fetches the data, creates a plot for the stock prices and moving averages, and saves the plot to a file.

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points): Date, Issue Description, Location of Resource, Result

Aya:
MarketAux API: Stock news articles API: MarketAux Documentation.
NLTK Library: Sentiment analysis using VADER: NLTK Documentation.
SQLite: Database for storing news articles and sentiment scores.
BeautifulSoup: For cleaning text extracted from API.
Matplotlib: For generating sentiment visualization plots.
Python Libraries: `requests`, `datetime`, `calendar`, and `sqlite3`.

Venkat:

Date	Issue Description	Location	Result
12/01	Needed a new financial website/API-owner to be able to retrieve metrics as previously selected website of not compatible for our needed.	https://finance.yahoo.com/	Success. Was able to use the API-wrappers that make API calls to Yahoo Finance to retrieve the newly selected metrics.
12/02	Needed some established Python library to be able to	https://matplotlib.org/stable/api/index.html	Success. Reference guide gave info all commands and

	graph and visualize multiple metrics and customize as needed.		method needed to create meaningful visuals.

Ricky:

Date	Issue Description	Location	Result
12/10/24	Needed to figure out a library/algorithm that can calculate the moving average of a set of numbers.	https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html	Success. Was able to calculate the moving average when given the window size.