# Bay atlantic university

# BGDA 501

**Assignment # 3**

Project API's & Data processing

**PROFESSOR:** Dr Mukul Sonwalker

April 17, 2025

**Submitted by:**

Gaurav Pandey

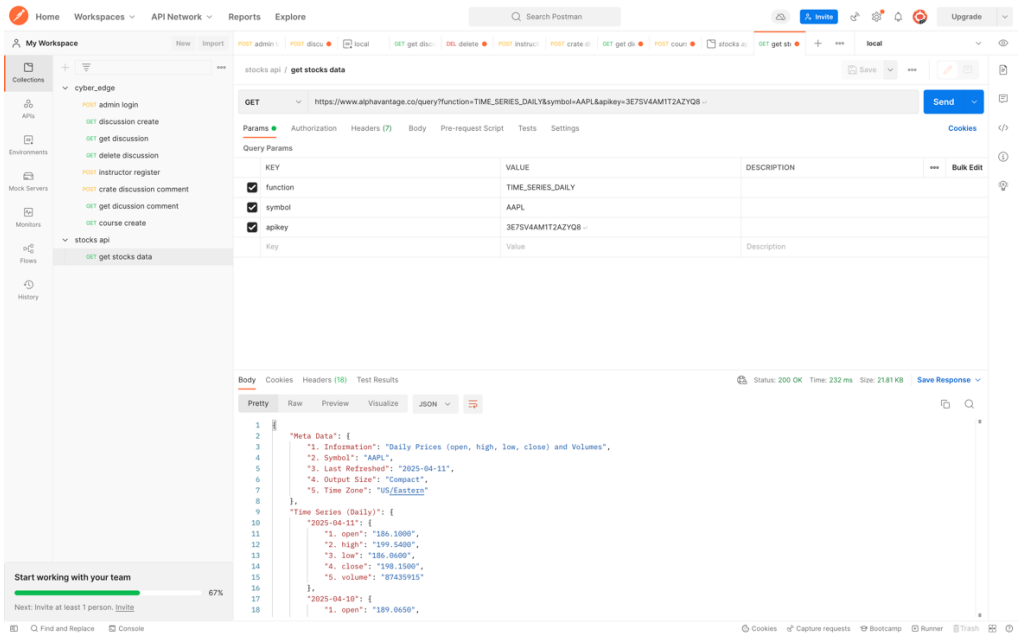
Meena Kasturi

Dovran Masharipov

At first, we choose the stocks api from Alpha vantage free api and then we generate the api key and follow the steps as the project asked:

1) Use an API testing tool (Ex. Postman) to perform a GET request to an API to review the data received by the sample, properly constructed API call

   ➔ We retrieve the stocks data of daily prices for open high low and close of daily time series.



```
    },
    "Time Series (Daily)": {
        "2025-04-11": {
            "1. open": "186.1000",
            "2. high": "199.5400",
            "3. low": "186.0600",
            "4. close": "198.1500",
            "5. volume": "87435915"
        },
        "2025-04-10": {
            "1. open": "189.0650",
            "2. high": "194.7799",
            "3. low": "183.0000",
            "4. close": "190.4200",
            "5. volume": "121879981"
        },
        "2025-04-09": {
            "1. open": "171.9500",
            "2. high": "200.6100",
            "3. low": "171.8900",
            "4. close": "198.8500",
            "5. volume": "184395885"
        },
        "2025-04-08": {
            "1. open": "186.7000",
            "2. high": "190.3350",
            "3. low": "169.2101",
            "4. close": "172.4200",
            "5. volume": "120859491"
        },
        "2025-04-07": {
            "1. open": "177.2000",
            "2. high": "194.1500",
            "3. low": "174.6200",
            "4. close": "181.4600",
            "5. volume": "160466286"
```


2) Use a Python script to make the API call and use a data frame to store the received results

➔ To make the api call we used alpha vantage for stocks data we need to create an api key we created it from this website - https://www.alphavantage.co/support/#api-key

We choose 5 stocks i.e. [ AAPL, GOOGL, MSFT, AMZN, TSLA]

Let us do for the AAPL stocks at first

```python
def fetch_data(symbol):
    url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=AAPL&apikey=3E7SV4AM1T2AZYQ8
    r = requests.get(url)
    data = r.json()
    time_series = data.get("Time Series (Daily)", {})
    df = pd.DataFrame.from_dict(time_series, orient='index').sort_index()
    df = df.rename(columns={
        '1. open': 'Open',
        '4. close': 'Close'
    }).astype(float)
    df['Price Change'] = df['Close'].diff()
    return df
```

We fetched the AAPL Stocks data for daily time frame from this script

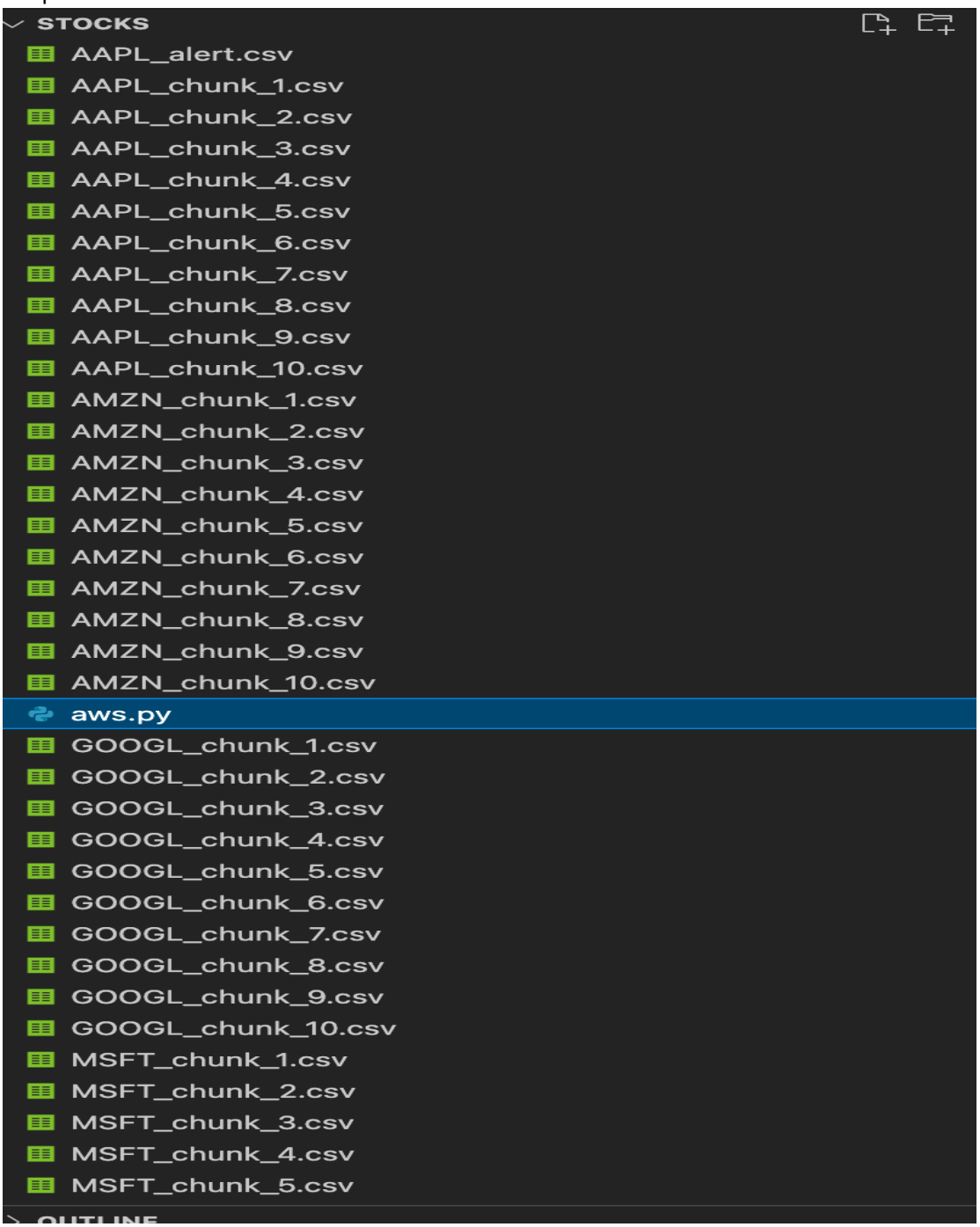
3) Parse the data stream (json) to split the data in 10-time steps i.e 20241115 – 20250411, etc. or based on 5 stock values

➔ As we can only call one stocks at a time in alpha vantage, we will parse the data steam and split the data in 10 dates of each stock individually as shown in the snapshot from 2024-11-15 to 2025-04-11

```python
import requests
import pandas as pd
import time
from datetime import datetime


api_key = '3E7SV4AM1T2AZYQ8'


stocks = ['AAPL', 'GOOGL', 'MSFT', 'AMZN', 'TSLA']

# Date range
start_date = '2024-11-15'
end_date = '2025-04-11'


def fetch_stock_data(symbol):
    url = f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=AAPL&apikey=3E7SV4AM1T2AZYQ
    response = requests.get(url)
    data = response.json().get("Time Series (Daily)", {})
    df = pd.DataFrame.from_dict(data, orient='index').sort_index()
    df = df.rename(columns={'4. close': 'Close'}).astype(float)
    df.index = pd.to_datetime(df.index)
    df = df.loc[(df.index >= start_date) & (df.index <= end_date)]
    return df[['Close']]


for symbol in stocks:
    print(f"Fetching: {symbol}")
    df = fetch_stock_data(symbol)

    if df.empty:
        print(f"No data found for {symbol}")
        continue
```

```python
    df = df.sort_index()
    chunks = []
    chunk_size = len(df) // 10

    for i in range(10):
        start = i * chunk_size
        end = (i + 1) * chunk_size if i < 9 else len(df)
        chunk_df = df.iloc[start:end]
        chunks.append(chunk_df)
        chunk_df.to_csv(f"{symbol}_chunk_{i+1}.csv")

    print(f"Split and saved 10 chunks for {symbol}")
    time.sleep(12)
```

Output:




10 chunks of each stocks is splitted and saved as a csv file.

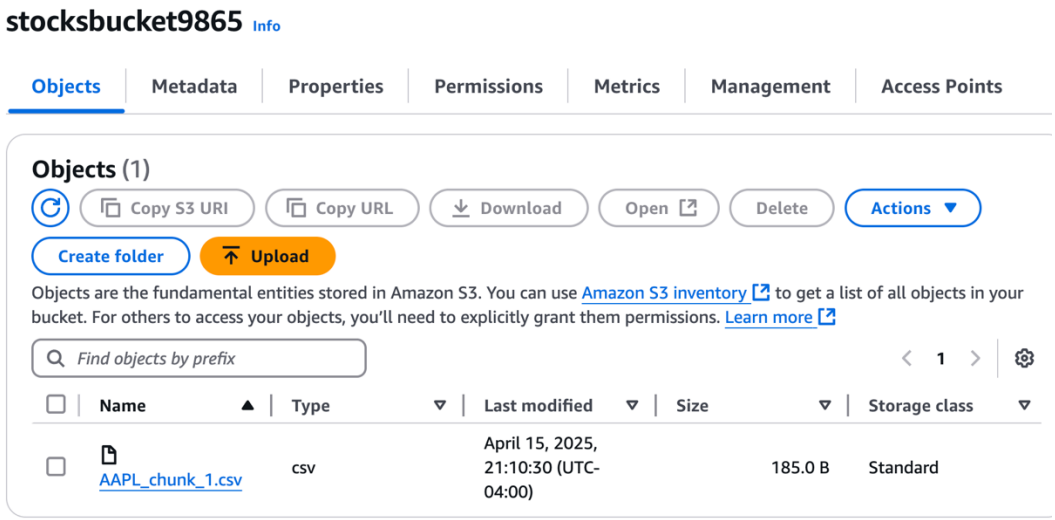
4) Pick any timestep (only one) and save the data (write to local disk) as a csv file and push to aws S3

➔ Now, to upload my csv file of all the stocks alerts at first, we created a s3 bucket which name is “stocksbucket9865”

After that using boto3 we uploaded our csv file as show in the snap shot

```python
import boto3

s3 = boto3.client('s3')

bucket_name = 'stocksbucket9865'

file_name = 'AAPL_chunk_1.csv'

try:
    s3.upload_file(file_name, bucket_name, file_name)
    print(f"✅ Uploaded 'AAPL_chunk_1' to S3 bucket 'stocksbucket9865'")
except Exception as e:
    print(f"Upload failed: {e}")
```

Output:




Similarly, we can upload other csv file to the s3 bucket by changing the files name in the code.