ASSIGNMENT-3
GLANCE-AT-FINANCE

BY VARADMURTY MOHOD (002772803)

ABOUT:

- A database consisting of live crypto and stock profiles of companies, crypto currencies, live news about stock profiles and other financial metrics for Algorithmic Trading, which can give investment insight for individuals and organizations looking to invest in different scales.

- GitHub Link: https://github.com/Varadmurty-mohod/GlanceAtFinance

DATA SOURCES:

- Most of the data is being pulled from the Alpha Vantage API directly and is being stored in a MySQL Database through a Jupyter Notebook.

- The API pulls include Stock Data, Crypto Currency Data, Foreign Exchange Data, Market News and Sentiment Data, Winning Portfolios, Earnings, IPO Calendar, Financial Metrics and much more.

- CSV inputs taken from Yahoo Finance.

SQL STATEMENTS:

- The following are the SQL statements for all the calls for data storage and tables.

CREATE DATABASE IF NOT EXISTS glance_at_finance;
CREATE TABLE IF NOT EXISTS daily_adjusted(
stock_name VARCHAR(100),
stock_day VARCHAR(20),
stock_low FLOAT,
stock_high FLOAT,
stock_open FLOAT,
stock_close FLOAT,
stock_volume FLOAT,
stock_dividend_amount FLOAT,
stock_split_coefficient FLOAT,

```sql
PRIMARY KEY (stock_day)
);

CREATE TABLE IF NOT EXISTS weekly(
stock_name VARCHAR(100),
stock_day VARCHAR(20),
stock_open FLOAT,
stock_high FLOAT,
stock_low FLOAT,
stock_close FLOAT,
stock_volume FLOAT,
PRIMARY KEY (stock_day)
);

CREATE TABLE IF NOT EXISTS monthly(
stock_name VARCHAR(100),
stock_day VARCHAR(20),
stock_open FLOAT,
stock_high FLOAT,
stock_low FLOAT,
stock_close FLOAT,
stock_volume FLOAT,
PRIMARY KEY (stock_day)
);

CREATE TABLE IF NOT EXISTS digital_currency_daily(
Currency_name VARCHAR(100),
Currency_day VARCHAR(20),
open_CNY FLOAT,
open_USD FLOAT,
high_CNY FLOAT,
high_USD FLOAT,
low_CNY FLOAT,
low_USD FLOAT,
close_CNY FLOAT,
close_USD FLOAT,
volume FLOAT,
market_cap_USD FLOAT,
PRIMARY KEY (Currency_day)
);

CREATE TABLE IF NOT EXISTS intraday(
stock_name VARCHAR(100),
stock_day VARCHAR(20),
stock_open FLOAT,
stock_high FLOAT,
stock_low FLOAT,
```

```sql
stock_close FLOAT,
stock_volume FLOAT,
PRIMARY KEY (stock_day)
);

CREATE TABLE IF NOT EXISTS earnings(
stock_name VARCHAR(100),
fiscalDateEnding VARCHAR(20),
reportedEPS FLOAT,
PRIMARY KEY (fiscalDateEnding)
);

CREATE TABLE IF NOT EXISTS fxexchange(
From_Currency VARCHAR(100),
To_Currency VARCHAR(100),
Exchange_Rate FLOAT,
PRIMARY KEY (From_Currency)
);

CREATE TABLE IF NOT EXISTS fxdaily(
FX_name VARCHAR(100),
FX_from VARCHAR(20),
FX_to VARCHAR(20),
FX_open FLOAT,
FX_high FLOAT,
FX_low FLOAT,
FX_close FLOAT,
PRIMARY KEY (FX_from)
);

CREATE TABLE IF NOT EXISTS digital_currency_exchange(
From_Currency VARCHAR(20),
To_Currency VARCHAR(20),
Exchange_Rate FLOAT,
PRIMARY KEY (From_Currency)
);

CREATE TABLE IF NOT EXISTS fxweekly(
FX_day VARCHAR(20),
FX_from VARCHAR(20),
FX_to VARCHAR(20),
FX_open FLOAT,
FX_high FLOAT,
FX_low FLOAT,
FX_close FLOAT,
PRIMARY KEY (FX_day)
);
```

```sql
CREATE TABLE IF NOT EXISTS digital_currency_weekly(
currency_name VARCHAR(100),
currency_day VARCHAR(20),
Market_Name VARCHAR(20),
open_CNY FLOAT,
open_USD FLOAT,
high_CNY FLOAT,
high_USD FLOAT,
low_CNY FLOAT,
low_USD FLOAT,
close_CNY FLOAT,
close_USD FLOAT,
volume FLOAT,
market_cap_USD FLOAT,
PRIMARY KEY (Currency_day)
);


CREATE TABLE IF NOT EXISTS digital_currency_monthly(
currency_name VARCHAR(100),
currency_day VARCHAR(20),
open_CNY FLOAT,
open_USD FLOAT,
high_CNY FLOAT,
high_USD FLOAT,
low_CNY FLOAT,
low_USD FLOAT,
close_CNY FLOAT,
close_USD FLOAT,
volume FLOAT,
market_cap_USD FLOAT,
PRIMARY KEY (Currency_day)
);

CREATE TABLE IF NOT EXISTS wma(
wma_name VARCHAR(100),
wma_day VARCHAR(20),
wma_value FLOAT,
PRIMARY KEY (wma_day)
);

CREATE TABLE IF NOT EXISTS dema(
dema_name VARCHAR(100),
dema_day VARCHAR(20),
dema_value FLOAT,
PRIMARY KEY (dema_day)
```

```sql
);

CREATE TABLE IF NOT EXISTS tema(
tema_name VARCHAR(100),
tema_day VARCHAR(20),
tema_value FLOAT,
PRIMARY KEY (tema_day)
);

CREATE TABLE IF NOT EXISTS trima(
trima_name VARCHAR(100),
trima_day VARCHAR(20),
trima_value FLOAT,
PRIMARY KEY (trima_day)
);

CREATE TABLE IF NOT EXISTS kama(
kama_name VARCHAR(100),
kama_day VARCHAR(20),
kama_value FLOAT,
PRIMARY KEY (kama_day)
);

CREATE TABLE IF NOT EXISTS mama(
mama_name VARCHAR(100),
mama_day VARCHAR(20),
mama_value FLOAT,
PRIMARY KEY (mama_day)
);

CREATE TABLE IF NOT EXISTS t3(
t3_name VARCHAR(100),
t3_day VARCHAR(20),
t3_value FLOAT,
PRIMARY KEY (t3_day)
);

CREATE TABLE IF NOT EXISTS macd(
macd_name VARCHAR(100),
macd_day VARCHAR(20),
MACD_Signal FLOAT,
MACD_Hist FLOAT,
MACD_value FLOAT,
PRIMARY KEY (macd_day)
);

CREATE TABLE IF NOT EXISTS stochf(
```

```sql
STOCHF_name VARCHAR(100),
STOCHF_day VARCHAR(20),
FastD FLOAT,
FastK FLOAT,
PRIMARY KEY (STOCHF_day)
);

CREATE TABLE IF NOT EXISTS stochrsi(
STOCHRSI_name VARCHAR(100),
STOCHRSI_day VARCHAR(20),
FastD FLOAT,
FastK FLOAT,
PRIMARY KEY (STOCHRSI_day)
);

CREATE TABLE IF NOT EXISTS willr(
WILLR_name VARCHAR(100),
WILLR_day VARCHAR(20),
WILLR_value FLOAT,
PRIMARY KEY (WILLR_day)
);

CREATE TABLE IF NOT EXISTS adxr(
ADXR_name VARCHAR(100),
ADXR_day VARCHAR(20),
ADXR_value FLOAT,
PRIMARY KEY (ADXR_day)
);

CREATE TABLE IF NOT EXISTS apo(
APO_name VARCHAR(100),
APO_day VARCHAR(20),
APO_value FLOAT,
PRIMARY KEY (APO_day)
);

CREATE TABLE IF NOT EXISTS ppo(
PPO_name VARCHAR(100),
PPO_day VARCHAR(20),
PPO_value FLOAT,
PRIMARY KEY (PPO_day)
);

CREATE TABLE IF NOT EXISTS mom(
MOM_name VARCHAR(100),
MOM_day VARCHAR(20),
MOM_value FLOAT,
```

```sql
PRIMARY KEY (MOM_day)
);

CREATE TABLE IF NOT EXISTS bop(
BOP_name VARCHAR(100),
BOP_day VARCHAR(20),
BOP_value FLOAT,
PRIMARY KEY (BOP_day)
);

CREATE TABLE IF NOT EXISTS cmo(
CMO_name VARCHAR(100),
CMO_day VARCHAR(20),
CMO_value FLOAT,
PRIMARY KEY (CMO_day)
);

CREATE TABLE IF NOT EXISTS roc(
ROC_name VARCHAR(100),
ROC_day VARCHAR(20),
ROC_value FLOAT,
PRIMARY KEY (ROC_day)
);

CREATE TABLE IF NOT EXISTS rocr(
ROCR_name VARCHAR(100),
ROCR_day VARCHAR(20),
ROCR_value FLOAT,
PRIMARY KEY (ROCR_day)
);

CREATE TABLE IF NOT EXISTS aroon(
AROON_name VARCHAR(100),
AROON_day VARCHAR(20),
Aroon_Up FLOAT,
Aroon_Down FLOAT,
PRIMARY KEY (AROON_day)
);

CREATE TABLE IF NOT EXISTS aroonosc(
AROONOSC_name VARCHAR(100),
AROONOSC_day VARCHAR(20),
AROONOSC_value FLOAT,
PRIMARY KEY (AROONOSC_day)
);

CREATE TABLE IF NOT EXISTS mfi(
```

```sql
MFI_name VARCHAR(100),
MFI_day VARCHAR(20),
MFI_value FLOAT,
PRIMARY KEY (MFI_day)
);

CREATE TABLE IF NOT EXISTS trix(
TRIX_name VARCHAR(100),
TRIX_day VARCHAR(20),
TRIX_value FLOAT,
PRIMARY KEY (TRIX_day)
);

CREATE TABLE IF NOT EXISTS ultosc(
ULTOSC_name VARCHAR(100),
ULTOSC_day VARCHAR(20),
ULTOSC_value FLOAT,
PRIMARY KEY (ULTOSC_day)
);

CREATE TABLE IF NOT EXISTS dx(
DX_name VARCHAR(100),
DX_day VARCHAR(20),
DX_value FLOAT,
PRIMARY KEY (DX_day)
);

CREATE TABLE IF NOT EXISTS minusdi(
MINUS_DI_name VARCHAR(100),
MINUS_DI_day VARCHAR(20),
MINUS_DI_value FLOAT,
PRIMARY KEY (MINUS_DI_day)
);

CREATE TABLE IF NOT EXISTS plusdi(
PLUS_DI_name VARCHAR(100),
PLUS_DI_day VARCHAR(20),
PLUS_DI_value FLOAT,
PRIMARY KEY (PLUS_DI_day)
);

CREATE TABLE IF NOT EXISTS minusdm(
MINUS_DM_name VARCHAR(100),
MINUS_DM_day VARCHAR(20),
MINUS_DM_value FLOAT,
PRIMARY KEY (MINUS_DM_day)
);
```

```sql
CREATE TABLE IF NOT EXISTS plusdm(
PLUS_DM_name VARCHAR(100),
PLUS_DM_day VARCHAR(20),
PLUS_DM_value FLOAT,
PRIMARY KEY (PLUS_DM_day)
);

CREATE TABLE IF NOT EXISTS midpoint(
MIDPOINT_name VARCHAR(100),
MIDPOINT_day VARCHAR(20),
MIDPOINT_value FLOAT,
PRIMARY KEY (MIDPOINT_day)
);

CREATE TABLE IF NOT EXISTS midprice(
MIDPRICE_name VARCHAR(100),
MIDPRICE_day VARCHAR(20),
MIDPRICE_value FLOAT,
PRIMARY KEY (MIDPRICE_day)
);

CREATE TABLE IF NOT EXISTS sar(
SAR_name VARCHAR(100),
SAR_day VARCHAR(20),
SAR_value FLOAT,
PRIMARY KEY (SAR_day)
);

CREATE TABLE IF NOT EXISTS trange(
TRANGE_name VARCHAR(100),
TRANGE_day VARCHAR(20),
TRANGE_value FLOAT,
PRIMARY KEY (TRANGE_day)
);

CREATE TABLE IF NOT EXISTS atr(
ATR_name VARCHAR(100),
ATR_day VARCHAR(20),
ATR_value FLOAT,
PRIMARY KEY (ATR_day)
);

CREATE TABLE IF NOT EXISTS Natr(
NATR_name VARCHAR(100),
NATR_day VARCHAR(20),
NATR_value FLOAT,
```

```
PRIMARY KEY (NATR_day)
);

CREATE TABLE IF NOT EXISTS ad(
AD_name VARCHAR(100),
AD_day VARCHAR(20),
AD_value FLOAT,
PRIMARY KEY (AD_day)
);

CREATE TABLE IF NOT EXISTS adosc(
ADOSC_name VARCHAR(100),
ADOSC_day VARCHAR(20),
ADOSC_value FLOAT,
PRIMARY KEY (ADOSC_day)
);

CREATE TABLE IF NOT EXISTS obv(
OBV_name VARCHAR(100),
OBV_day VARCHAR(20),
OBV_value FLOAT,
PRIMARY KEY (OBV_day)
);

CREATE TABLE IF NOT EXISTS httrendline(
HT_TRENDLINE_name VARCHAR(100),
HT_TRENDLINE_day VARCHAR(20),
HT_TRENDLINE_value FLOAT,
PRIMARY KEY (HT_TRENDLINE_day)
);

CREATE TABLE IF NOT EXISTS htsine(
HT_SINE_name VARCHAR(100),
HT_SINE_day VARCHAR(20),
LEAD_SINE FLOAT,
SINE FLOAT,
PRIMARY KEY (HT_SINE_day)
);

CREATE TABLE IF NOT EXISTS httrendmode(
HT_TRENDMODE_name VARCHAR(100),
HT_TRENDMODE_day VARCHAR(20),
HT_TRENDMODE_VALUE FLOAT,
PRIMARY KEY (HT_TRENDMODE_day)
);

CREATE TABLE IF NOT EXISTS htdcperiod(
```
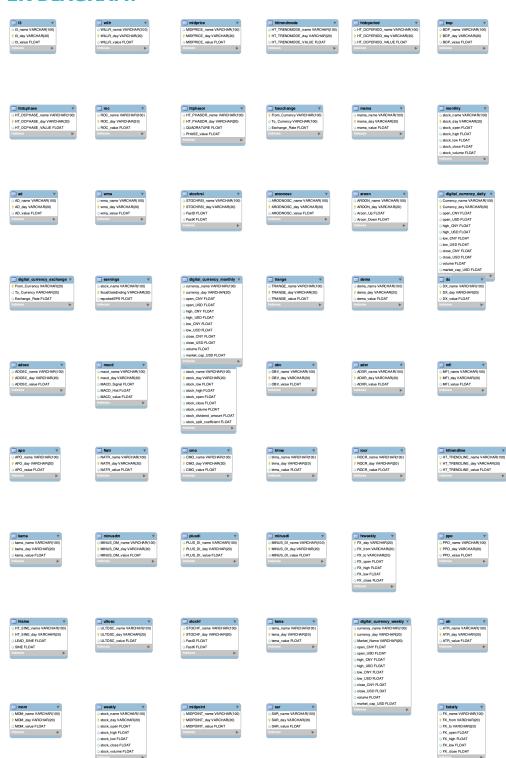
```sql
HT_DCPERIOD_name VARCHAR(100),
HT_DCPERIOD_day VARCHAR(20),
HT_DCPERIOD_VALUE FLOAT,
PRIMARY KEY (HT_DCPERIOD_day)
);

CREATE TABLE IF NOT EXISTS htdcphase(
HT_DCPHASE_name VARCHAR(100),
HT_DCPHASE_day VARCHAR(20),
HT_DCPHASE_VALUE FLOAT,
PRIMARY KEY (HT_DCPHASE_day)
);

CREATE TABLE IF NOT EXISTS htphasor(
HT_PHASOR_name VARCHAR(100),
HT_PHASOR_day VARCHAR(20),
QUADRATURE FLOAT,
PHASE_value FLOAT,
PRIMARY KEY (HT_PHASOR_day)
);
```

JUPYTER NOTEBOOK:

- The following is the print export of the Jupyter Notebook with all the Data Processing and commands for storage linked to MySQL.

  PFA AT THE END OF THE DOC.

ER DIAGRAM:

t3
- t3_name VARCHAR(100)
- t3_day VARCHAR(20)
- t3_value FLOAT

willr
- WILLR_name VARCHAR(100)
- WILLR_day VARCHAR(20)
- WILLR_value FLOAT

midprice
- MIDPRICE_name VARCHAR(100)
- MIDPRICE_day VARCHAR(20)
- MIDPRICE_value FLOAT

httrendmode
- HT_TRENDMODE_name VARCHAR(100)
- HT_TRENDMODE_day VARCHAR(20)
- HT_TRENDMODE_VALUE FLOAT

htdcperiod
- HT_DCPERIOD_name VARCHAR(100)
- HT_DCPERIOD_day VARCHAR(20)
- HT_DCPERIOD_VALUE FLOAT

bop
- BOP_name VARCHAR(100)
- BOP_day VARCHAR(20)
- BOP_value FLOAT

htdcphase
- HT_DCPHASE_name VARCHAR(100)
- HT_DCPHASE_day VARCHAR(20)
- HT_DCPHASE_VALUE FLOAT

roc
- ROC_name VARCHAR(100)
- ROC_day VARCHAR(20)
- ROC_value FLOAT

htphasor
- HT_PHASOR_name VARCHAR(100)
- HT_PHASOR_day VARCHAR(20)
- QUADRATURE FLOAT
- PHASE_value FLOAT

fxexchange
- From_Currency VARCHAR(100)
- To_Currency VARCHAR(100)
- Exchange_Rate FLOAT

mama
- mama_name VARCHAR(100)
- mama_day VARCHAR(20)
- mama_value FLOAT

monthly
- stock_name VARCHAR(100)
- stock_day VARCHAR(20)
- stock_open FLOAT
- stock_high FLOAT
- stock_low FLOAT
- stock_close FLOAT
- stock_volume FLOAT

ad
- AD_name VARCHAR(100)
- AD_day VARCHAR(20)
- AD_value FLOAT

wma
- wma_name VARCHAR(100)
- wma_day VARCHAR(20)
- wma_value FLOAT

stochrsi
- STOCHRSI_name VARCHAR(100)
- STOCHRSI_day VARCHAR(20)
- FastD FLOAT
- FastK FLOAT

aroonosc
- AROONOSC_name VARCHAR(100)
- AROONOSC_day VARCHAR(20)
- AROONOSC_value FLOAT

aroon
- AROON_name VARCHAR(100)
- AROON_day VARCHAR(20)
- Aroon_Up FLOAT
- Aroon_Down FLOAT

digital_currency_daily
- Currency_name VARCHAR(100)
- Currency_day VARCHAR(20)
- open_CNY FLOAT
- open_USD FLOAT
- high_CNY FLOAT
- high_USD FLOAT
- low_CNY FLOAT
- low_USD FLOAT
- close_CNY FLOAT
- close_USD FLOAT
- volume FLOAT
- market_cap_USD FLOAT

digital_currency_exchange
- From_Currency VARCHAR(20)
- To_Currency VARCHAR(20)
- Exchange_Rate FLOAT

earnings
- stock_name VARCHAR(100)
- fiscalDateEnding VARCHAR(20)
- reportedEPS FLOAT

digital_currency_monthly
- currency_name VARCHAR(100)
- currency_day VARCHAR(20)
- open_CNY FLOAT
- open_USD FLOAT
- high_CNY FLOAT
- high_USD FLOAT
- low_CNY FLOAT
- low_USD FLOAT
- close_CNY FLOAT
- close_USD FLOAT
- volume FLOAT
- market_cap_USD FLOAT

trange
- TRANGE_name VARCHAR(100)
- TRANGE_day VARCHAR(20)
- TRANGE_value FLOAT

dema
- dema_name VARCHAR(100)
- dema_day VARCHAR(20)
- dema_value FLOAT

dx
- DX_name VARCHAR(100)
- DX_day VARCHAR(20)
- DX_value FLOAT

(table continued)
- stock_name VARCHAR(100)
- stock_day VARCHAR(20)
- stock_low FLOAT
- stock_high FLOAT
- stock_open FLOAT
- stock_close FLOAT
- stock_volume FLOAT
- stock_dividend_amount FLOAT
- stock_split_coefficient FLOAT

adosc
- ADOSC_name VARCHAR(100)
- ADOSC_day VARCHAR(20)
- ADOSC_value FLOAT

macd
- macd_name VARCHAR(100)
- macd_day VARCHAR(20)
- MACD_Signal FLOAT
- MACD_Hist FLOAT
- MACD_value FLOAT

obv
- OBV_name VARCHAR(100)
- OBV_day VARCHAR(20)
- OBV_value FLOAT

adxr
- ADXR_name VARCHAR(100)
- ADXR_day VARCHAR(20)
- ADXR_value FLOAT

mfi
- MFI_name VARCHAR(100)
- MFI_day VARCHAR(20)
- MFI_value FLOAT

apo
- APO_name VARCHAR(100)
- APO_day VARCHAR(20)
- APO_value FLOAT

Natr
- NATR_name VARCHAR(100)
- NATR_day VARCHAR(20)
- NATR_value FLOAT

cmo
- CMO_name VARCHAR(100)
- CMO_day VARCHAR(20)
- CMO_value FLOAT

trima
- trima_name VARCHAR(100)
- trima_day VARCHAR(20)
- trima_value FLOAT

rocr
- ROCR_name VARCHAR(100)
- ROCR_day VARCHAR(20)
- ROCR_value FLOAT

httrendline
- HT_TRENDLINE_name VARCHAR(100)
- HT_TRENDLINE_day VARCHAR(20)
- HT_TRENDLINE_value FLOAT

kama
- kama_name VARCHAR(100)
- kama_day VARCHAR(20)
- kama_value FLOAT

minusdm
- MINUS_DM_name VARCHAR(100)
- MINUS_DM_day VARCHAR(20)
- MINUS_DM_value FLOAT

plusdi
- PLUS_DI_name VARCHAR(100)
- PLUS_DI_day VARCHAR(20)
- PLUS_DI_value FLOAT

minusdi
- MINUS_DI_name VARCHAR(100)
- MINUS_DI_day VARCHAR(20)
- MINUS_DI_value FLOAT

fxweekly
- FX_day VARCHAR(20)
- FX_from VARCHAR(20)
- FX_to VARCHAR(20)
- FX_open FLOAT
- FX_high FLOAT
- FX_low FLOAT
- FX_close FLOAT

ppo
- PPO_name VARCHAR(100)
- PPO_day VARCHAR(20)
- PPO_value FLOAT

htsine
- HT_SINE_name VARCHAR(100)
- HT_SINE_day VARCHAR(20)
- LEAD_SINE FLOAT
- SINE FLOAT

ultosc
- ULTOSC_name VARCHAR(100)
- ULTOSC_day VARCHAR(20)
- ULTOSC_value FLOAT

stochf
- STOCHF_name VARCHAR(100)
- STOCHF_day VARCHAR(20)
- FastD FLOAT
- FastK FLOAT

tema
- tema_name VARCHAR(100)
- tema_day VARCHAR(20)
- tema_value FLOAT

digital_currency_weekly
- currency_name VARCHAR(100)
- currency_day VARCHAR(20)
- Market_Name VARCHAR(20)
- open_CNY FLOAT
- open_USD FLOAT
- high_CNY FLOAT
- high_USD FLOAT
- low_CNY FLOAT
- low_USD FLOAT
- close_CNY FLOAT
- close_USD FLOAT
- volume FLOAT
- market_cap_USD FLOAT

atr
- ATR_name VARCHAR(100)
- ATR_day VARCHAR(20)
- ATR_value FLOAT

mom
- MOM_name VARCHAR(100)
- MOM_day VARCHAR(20)
- MOM_value FLOAT

weekly
- stock_name VARCHAR(100)
- stock_day VARCHAR(20)
- stock_open FLOAT
- stock_high FLOAT
- stock_low FLOAT
- stock_close FLOAT
- stock_volume FLOAT

midpoint
- MIDPOINT_name VARCHAR(100)
- MIDPOINT_day VARCHAR(20)
- MIDPOINT_value FLOAT

sar
- SAR_name VARCHAR(100)
- SAR_day VARCHAR(20)
- SAR_value FLOAT

fxdaily
- FX_name VARCHAR(100)
- FX_from VARCHAR(20)
- FX_to VARCHAR(20)
- FX_open FLOAT
- FX_high FLOAT
- FX_low FLOAT
- FX_close FLOAT

trix
- TRIX_name VARCHAR(100)
- TRIX_day VARCHAR(20)
- TRIX_value FLOAT

plusdm
- PLUS_DM_name VARCHAR(100)
- PLUS_DM_day VARCHAR(20)
- PLUS_DM_value FLOAT

QUERY STATEMENTS:

1. Show me the Negative and Positive Directional indication for the stock AAPL where the variance is greater than 20 for Negative DI and over 30 for positive DI.

- -DI is part of a more comprehensive indicator called the Average Directional Index (ADX). The ADX reveals trend direction and trend strength.

- The indicator was designed by Welles Wilder for commodities, it is used for other markets and on all timeframes.1

- When the Negative Directional Indicator (-DI) moves up and is above the Positive Directional Indicator (+DI), then the price downtrend is getting stronger.

- When -DI is moving down, and below the +DI, then the price uptrend is strengthening.

- When +DI and -DI crossover, it indicates the possibility of a new trend. If -DI crosses above the +DI, then a new downtrend could be starting.

SELECT

  t1.MINUS_DI_name, t1.MINUS_DI_value, t2.PLUS_DI_value

FROM

  minusdi t1

    INNER JOIN

  plusdi t2 ON MINUS_DI_name = PLUS_DI_name

WHERE

  MINUS_DI_value > 20

    AND PLUS_DI_value > 30;

2. Show me the AROON value with their existing oscillation values for the stock AAPL.

- The Aroon indicator is composed of two lines. An up line which measures the number of periods since a High, and a down line which measures the number of periods since a Low.

- The indicator is typically applied to 25 periods of data, so the indicator is showing how many periods it has been since a 25-period high or low.

- When the Aroon Up is above the Aroon Down, it indicates bullish price behavior.

- When the Aroon Down is above the Aroon Up, it signals bearish price behavior.

- Crossovers of the two lines can signal trend changes. For example, when Aroon Up crosses above Aroon Down it may mean a new uptrend is starting.

- The indicator moves between zero and 100. A reading above 50 means that a high/low (whichever line is above 50) was seen within the last 12 periods.

- A reading below 50 means that the high/low was seen within the 13 periods.

- The Aroon Oscillator uses Aroon Up and Aroon Down to create the oscillator.

- Aroon Up and Aroon Down measure the number of periods since the last 25-period high and low.

- The Aroon Oscillator crosses above the zero line when Aroon Up moves above Aroon Down. The oscillator drops below the zero line when the Aroon Down moves below the Aroon Up.

```
SELECT
  t1.AROON_name, t1.Aroon_Up, t1.Aroon_Down, t2.AROONOSC_value
FROM
  aroon t1
    INNER JOIN
  aroonosc t2 ON AROON_name = AROONOSC_name
  WHERE AROONOSC_value>0;
```

3. Show me all the Hilbert Transform values available.

- Created by John Ehlers, the Hilbert Transform is a 5-period trendline of high/low price that that uses classic electrical radio-frequency signal processing algorithms reduce noise.

```
SELECT
  t1.HT_TRENDLINE_name, t1.HT_TRENDLINE_value, t2.LEAD_SINE, t2.SINE,
t3.HT_TRENDMODE_VALUE, t4.HT_DCPERIOD_VALUE, t5.HT_DCPHASE_VALUE,
t6.QUADRATURE, t6.PHASE_value
FROM
```

httrendline t1

    INNER join htsine t2 ON HT_TRENDLINE_name = HT_SINE_name

    inner join httrendmode t3 ON HT_TRENDMODE_name = HT_TRENDLINE_name

    inner join htdcperiod t4 ON HT_DCPERIOD_name = HT_TRENDLINE_name

    inner join htdcphase t5 ON HT_DCPHASE_name = HT_TRENDLINE_name

       inner join htphasor t6 ON HT_DCPHASE_name = HT_TRENDLINE_name;

4. Show me the historical time series for Bitcoin with its exchange rate, when it was in profit at the end of the trading day.

SELECT

  *

FROM

  digital_currency_exchange t1

    INNER JOIN

  digital_currency_weekly t2 ON From_Currency = currency_name

    AND To_Currency = Market_Name

    WHERE close_USD>open_USD;

5. Show me all the possible moving average values for the stock AAPL.

SELECT

  t1.wma_value, t2.dema_value, t3.tema_value, t4.trima_value, t5.kama_value, t6.mama_value

FROM

  wma t1

  inner join dema t2 ON wma_name = dema_name

  inner join tema t3 ON dema_name = wma_name

  inner join trima t4 ON tema_name = wma_name

  inner join kama t5 ON trima_name = wma_name

  inner join mama t6 ON kama_name = wma_name;

```python
from alpha_vantage.timeseries import TimeSeries
from alpha_vantage.foreignexchange import ForeignExchange
from alpha_vantage.cryptocurrencies import CryptoCurrencies
from alpha_vantage.techindicators import TechIndicators
from alpha_vantage.sectorperformance import SectorPerformances
import os
import json
import requests
import pandas as pd


app = TimeSeries('R8QBN54GF80WJUT6')
```

```python
# Example print
aapl = app.get_daily_adjusted('AAPL', outputsize='full')
print(json.dumps(aapl, indent=2))
```

```python
import mysql.connector
import os

# establishing the connection and creating cursor
try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')
```

```python
#search function
url = 'https://www.alphavantage.co/query?function=SYMBOL_SEARCH&keywords=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()
```

```python
# DAILY ADJUSTED

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# Daily Adjusted

url = 'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

timeSeries = data["Time Series (Daily)"]

for x in timeSeries:
    stock_day = x
    stock_name = data["Meta Data"]["2. Symbol"]
    stock_open = timeSeries[x]["1. open"]
    stock_high = timeSeries[x]["2. high"]
    stock_low = timeSeries[x]["3. low"]
    stock_close = timeSeries[x]["4. close"]
    stock_volume = timeSeries[x]["6. volume"]
    stock_dividend_amount = timeSeries[x]["7. dividend amount"]
    stock_split_coefficient = timeSeries[x]["8. split coefficient"]
    try:
        query = "INSERT INTO daily_adjusted(stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume, stock_dividend_amount, stock_split_coefficient) " \
                "VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        args = (stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume, stock_dividend_amount, stock_split_coefficient)

        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# WEEKLY

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# Weekly
url = 'https://www.alphavantage.co/query?function=TIME_SERIES_WEEKLY&symbol=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

timeSeries = data["Weekly Time Series"]

for x in timeSeries:
    stock_day = x
    stock_name = data["Meta Data"]["2. Symbol"]
    stock_open = timeSeries[x]["1. open"]
    stock_high = timeSeries[x]["2. high"]
    stock_low = timeSeries[x]["3. low"]
    stock_close = timeSeries[x]["4. close"]
    stock_volume = timeSeries[x]["5. volume"]
    try:
        query = "INSERT INTO weekly(stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s)"
        args = (stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)

        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')
# print(json.dumps(data, indent=2))
```

```python
# MONTHLY

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# Monthly
url = 'https://www.alphavantage.co/query?function=TIME_SERIES_MONTHLY&symbol=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

timeSeries = data["Monthly Time Series"]

for x in timeSeries:
    stock_day = x
    stock_name = data["Meta Data"]["2. Symbol"]
    stock_open = timeSeries[x]["1. open"]
    stock_high = timeSeries[x]["2. high"]
    stock_low = timeSeries[x]["3. low"]
    stock_close = timeSeries[x]["4. close"]
    stock_volume = timeSeries[x]["5. volume"]
    try:
        query = "INSERT INTO monthly(stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s)"
        args = (stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)

        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')
# print(json.dumps(data, indent=2))
```

```python
# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# Digital Currency Daily
url = 'https://www.alphavantage.co/query?function=DIGITAL_CURRENCY_DAILY&symbol=BTC&market=CNY&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

timeSeries = data["Time Series (Digital Currency Daily)"]

for x in timeSeries:
    Currency_day = x
    Currency_name = data["Meta Data"]["3. Digital Currency Name"]
    open_CNY = timeSeries[x]["1a. open (CNY)"]
    open_USD = timeSeries[x]["1b. open (USD)"]
    high_CNY = timeSeries[x]["2a. high (CNY)"]
    high_USD = timeSeries[x]["2b. high (USD)"]
    low_CNY = timeSeries[x]["3a. low (CNY)"]
    low_USD = timeSeries[x]["3b. low (USD)"]
    close_CNY = timeSeries[x]["4a. close (CNY)"]
    close_USD = timeSeries[x]["4b. close (USD)"]
    volume = timeSeries[x]["5. volume"]
    market_cap_USD = timeSeries[x]["6. market cap (USD)"]
    try:
        query = "INSERT INTO digital_currency_daily(Currency_day, Currency_name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        args = (Currency_day, Currecncy_name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# print(json.dumps(data, indent=2))
```

```python
# connection to MySql

# INTRADAY

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

# Daily Adjusted

url = 'https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=AAPL&interval=5min&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))

timeSeries = data["Time Series (5min)"]

for x in timeSeries:
    stock_day = x
    stock_name = data["Meta Data"]["2. Symbol"]
    stock_open = timeSeries[x]["1. open"]
    stock_high = timeSeries[x]["2. high"]
    stock_low = timeSeries[x]["3. low"]
    stock_close = timeSeries[x]["4. close"]
    stock_volume = timeSeries[x]["5. volume"]
    try:
        query = "INSERT INTO intraday(stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s)"
        args = (stock_name, stock_day, stock_high, stock_open, stock_low, stock_close, stock_volume)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")
```

```python
# Market News & Sentiment
# This API returns live and historical market news & sentiment data derived from over 50 major financial news outlets around the world, covering stocks, cryptocurrencies, forex, and a wide range of topics such as fis
import requests

url = 'https://www.alphavantage.co/query?function=NEWS_SENTIMENT&tickers=AAPL&topics=technology&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# Winning Portfolios
# This API returns the historical portfolio rankings from the Alpha Tournament, world's leading portfolio competition and investors community.

import requests

url = 'https://www.alphavantage.co/query?function=TOURNAMENT_PORTFOLIO&season=2021-09&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# COMPANY REVIEW

import requests

url = 'https://www.alphavantage.co/query?function=OVERVIEW&symbol=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# EARNINGS
# This API returns the annual and quarterly earnings (EPS) for the company of interest. Quarterly data also includes analyst estimates and surprise metrics.


# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

import requests

url = 'https://www.alphavantage.co/query?function=EARNINGS&symbol=AAPL&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

# print(json.dumps(data, indent=2))

annualEarnings = data["annualEarnings"]

for x, y in annualEarnings:
    fiscalDateEnding = x
    reportedEPS = y
    stock_name = data["symbol"]

    try:
        query = "INSERT INTO earnings(stock_name, fiscalDateEnding, reportedEPS)" \
                "VALUES(%s,%s,%s)"
        args = (stock_name, fiscalDateEnding, reportedEPS)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")
```

```python
# Listing & Delisting Status
# Returns a list of active or delisted US stocks and ETFs, either as of the latest trading day or at a specific time in history. The endpoint is positioned to facilitate equity research on asset lifecycle and surviv

import csv
import requests

CSV_URL = 'https://www.alphavantage.co/query?function=LISTING_STATUS&apikey=R8QBN54GF80WJUT6'

with requests.Session() as s:
    download = s.get(CSV_URL)
    decoded_content = download.content.decode('utf-8')
    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    my_list = list(cr)
    for row in my_list:
        print(row)
```

```python
# Earnings Calendar
# This API returns a list of company earnings expected in the next 3, 6, or 12 months.


import csv
import requests

# replace the "demo" apikey below with your own key from https://www.alphavantage.co/support/#api-key
CSV_URL = 'https://www.alphavantage.co/query?function=EARNINGS_CALENDAR&horizon=3month&apikey=R8QBN54GF80WJUT6'

with requests.Session() as s:
    download = s.get(CSV_URL)
    decoded_content = download.content.decode('utf-8')
    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    my_list = list(cr)
    for row in my_list:
        print(row)
```

```python
# IPO Calendar
# This API returns a list of IPOs expected in the next 3 months.

import csv
import requests

# replace the "demo" apikey below with your own key from https://www.alphavantage.co/support/#api-key
CSV_URL = 'https://www.alphavantage.co/query?function=IPO_CALENDAR&apikey=R8QBN54GF80WJUT6'

with requests.Session() as s:
    download = s.get(CSV_URL)
    decoded_content = download.content.decode('utf-8')
    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    my_list = list(cr)
    for row in my_list:
        print(row)
```

```python
# CURRENCY_EXCHANGE_RATE
# Returns the realtime exchange rate for a pair of digital currency (e.g., Bitcoin) and physical currency (e.g., USD).

import requests

url = 'https://www.alphavantage.co/query?function=CURRENCY_EXCHANGE_RATE&from_currency=USD&to_currency=JPY&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()


    From_Currency = data["Realtime Currency Exchange Rate"]["2. From_Currency Name"]
    To_Currency = data["Realtime Currency Exchange Rate"]["4. To_Currency Name"]
    Exchange_Rate = data["Realtime Currency Exchange Rate"]["5. Exchange Rate"]


    try:
        query = "INSERT INTO fxexchange(From_Currency, To_Currency, Exchange_Rate)" \
                "VALUES(%s,%s,%s)"
        args = (From_Currency, To_Currency, Exchange_Rate)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')


print(json.dumps(data, indent=2))
```

```python
# FX_DAILY
# This API returns the daily time series (timestamp, open, high, low, close) of the FX currency pair specified, updated realtime.

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

import requests

url = 'https://www.alphavantage.co/query?function=FX_DAILY&from_symbol=EUR&to_symbol=USD&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()


timeSeries = data["Time Series FX (Daily)"]

for x in timeSeries:
    FX_day = x
    FX_from = data["Meta Data"]["2. From Symbol"]
    FX_to = data["Meta Data"]["3. To Symbol"]
    FX_open = timeSeries[x]["1. open"]
    FX_high = timeSeries[x]["2. high"]
    FX_low = timeSeries[x]["3. low"]
    FX_close = timeSeries[x]["4. close"]
    try:
        query = "INSERT INTO fxdaily(FX_from, FX_to, FX_day, FX_high, FX_open, FX_low, FX_close)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s)"
        args = (FX_from, FX_to, FX_day, FX_high, FX_open, FX_low, FX_close)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')


# print(json.dumps(data, indent=2))
```

```python
# FX_WEEKLY
# Returns the weekly time series (timestamp, open, high, low, close) of the FX currency pair specified, updated realtime.The latest data point is the price information for the week (or partial week) containing the cu

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

import requests

url = 'https://www.alphavantage.co/query?function=FX_WEEKLY&from_symbol=EUR&to_symbol=USD&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

# print(json.dumps(data, indent=2))

timeSeries = data["Time Series FX (Weekly)"]

for x in timeSeries:
    FX_day = x
    FX_from = data["Meta Data"]["2. From Symbol"]
    FX_to = data["Meta Data"]["3. To Symbol"]
    FX_open = timeSeries[x]["1. open"]
    FX_high = timeSeries[x]["2. high"]
    FX_low = timeSeries[x]["3. low"]
    FX_close = timeSeries[x]["4. close"]
    try:
        query = "INSERT INTO fxweekly(FX_day, FX_from, FX_to, FX_high, FX_open, FX_low, FX_close)" \
                "VALUES(%s,%s,%s,%s,%s,%s,%s)"
        args = (FX_day, FX_from, FX_to, FX_high, FX_open, FX_low, FX_close)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")
```

```python
# FX_MONTHLY
# Returns the monthly time series (timestamp, open, high, low, close) of the FX currency pair specified, updated realtime.The latest data point is the prices information for the month (or partial month) containing th

import requests

url = 'https://www.alphavantage.co/query?function=FX_MONTHLY&from_symbol=EUR&to_symbol=USD&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# CURRENCY_EXCHANGE_RATE
# This API returns the realtime exchange rate for any pair of digital currency (e.g., Bitcoin) or physical currency (e.g., USD).

import requests

url = 'https://www.alphavantage.co/query?function=CURRENCY_EXCHANGE_RATE&from_currency=BTC&to_currency=CNY&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

# connection to MySql
import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

rcer = data["Realtime Currency Exchange Rate"]

From_Currency = rcer["2. From_Currency Name"]
To_Currency = rcer["4. To_Currency Name"]
Exchange_Rate = rcer["5. Exchange Rate"]

try:
    query = "INSERT INTO digital_currency_exchange(From_Currency, To_Currency, Exchange_Rate)" \
            "VALUES(%s,%s,%s)"
    args = (From_Currency, To_Currency, Exchange_Rate)
    conn.cursor().execute(query, args)
    conn.commit()
except:
    print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')


# print(json.dumps(data, indent=2))
```

```
In [ ]:  # DIGITAL_CURRENCY_WEEKLY
         # Returns the weekly historical time series for a digital currency (e.g., BTC) traded on a specific market (e.g., CNY/Chinese Yuan), refreshed daily at midnight (UTC). Prices and volumes are quoted in both the market

         import requests

         url = 'https://www.alphavantage.co/query?function=DIGITAL_CURRENCY_WEEKLY&symbol=BTC&market=CNY&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         # connection to MySql
         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Time Series (Digital Currency Weekly)"]

         for x in timeSeries:
             currency_day = x
             currency_name = data["Meta Data"]["3. Digital Currency Name"]
             Market_Name = data["Meta Data"]["5. Market Name"]
             open_CNY = timeSeries[x]["1a. open (CNY)"]
             open_USD = timeSeries[x]["1b. open (USD)"]
             high_CNY = timeSeries[x]["2a. high (CNY)"]
             high_USD = timeSeries[x]["2b. high (USD)"]
             low_CNY = timeSeries[x]["3a. low (CNY)"]
             low_USD = timeSeries[x]["3b. low (USD)"]
             close_CNY = timeSeries[x]["4a. close (CNY)"]
             close_USD = timeSeries[x]["4b. close (USD)"]
             volume = timeSeries[x]["5. volume"]
             market_cap_USD = timeSeries[x]["6. market cap (USD)"]
             try:
                 query = "INSERT INTO digital_currency_weekly(currency_day, currency_name, Market_Name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)" \
                         "VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
                 args = (currency_day, currency_name, Market_Name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')


         print(json.dumps(data, indent=2))

In [ ]:  # DIGITAL_CURRENCY_MONTHLY
         # Returns the monthly historical time series for a digital currency (e.g., BTC) traded on a specific market (e.g., CNY/Chinese Yuan), refreshed daily at midnight (UTC). Prices and volumes are quoted in both the marke

         import requests

         url = 'https://www.alphavantage.co/query?function=DIGITAL_CURRENCY_MONTHLY&symbol=BTC&market=CNY&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         # connection to MySql
         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Time Series (Digital Currency Monthly)"]

         for x in timeSeries:
             currency_day = x
             currency_name = data["Meta Data"]["3. Digital Currency Name"]
             open_CNY = timeSeries[x]["1a. open (CNY)"]
             open_USD = timeSeries[x]["1b. open (USD)"]
             high_CNY = timeSeries[x]["2a. high (CNY)"]
             high_USD = timeSeries[x]["2b. high (USD)"]
             low_CNY = timeSeries[x]["3a. low (CNY)"]
             low_USD = timeSeries[x]["3b. low (USD)"]
             close_CNY = timeSeries[x]["4a. close (CNY)"]
             close_USD = timeSeries[x]["4b. close (USD)"]
             volume = timeSeries[x]["5. volume"]
             market_cap_USD = timeSeries[x]["6. market cap (USD)"]
             try:
                 query = "INSERT INTO digital_currency_monthly(currency_day, currency_name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)" \
                         "VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
                 args = (currency_day, currency_name, open_CNY, open_USD, high_CNY, high_USD, low_CNY, low_USD, close_CNY, close_USD, volume, market_cap_USD)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```python
# REAL_GDP
# Returns the annual and quarterly Real GDP of the United States.

import requests

url = 'https://www.alphavantage.co/query?function=REAL_GDP&interval=annual&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# REAL_GDP_PER_CAPITA
# Returns the quarterly Real GDP per Capita data of the United States.

import requests

url = 'https://www.alphavantage.co/query?function=REAL_GDP_PER_CAPITA&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# TREASURY_YIELD
# Returns the daily, weekly, and monthly US treasury yield of a given maturity timeline (e.g., 5 year, 30 year, etc).

import requests

url = 'https://www.alphavantage.co/query?function=TREASURY_YIELD&interval=monthly&maturity=10year&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# FEDERAL_FUNDS_RATE
# Returns the daily, weekly, and monthly federal funds rate (interest rate) of the United States.

import requests

url = 'https://www.alphavantage.co/query?function=FEDERAL_FUNDS_RATE&interval=monthly&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# CPI
# Returns the monthly and semiannual consumer price index (CPI) of the United States. CPI is widely regarded as the barometer of inflation levels in the broader economy.

import requests

url = 'https://www.alphavantage.co/query?function=CPI&interval=monthly&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# INFLATION
# Returns the annual inflation rates (consumer prices) of the United States.

import requests

url = 'https://www.alphavantage.co/query?function=INFLATION&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# INFLATION_EXPECTATION
# Returns the monthly inflation expectation data of the United States, as measured by the median expected price change next 12 months according to the Surveys of Consumers by University of Michigan (Inflation Expecta

import requests

url = 'https://www.alphavantage.co/query?function=INFLATION_EXPECTATION&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# CONSUMER_SENTIMENT
# Returns the monthly consumer sentiment and confidence data of the United States, as measured by the Surveys of Consumers by University of Michigan (Consumer Sentiment © [UMCSENT]), retrieved from FRED, Federal Rese

import requests

url = 'https://www.alphavantage.co/query?function=CONSUMER_SENTIMENT&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```python
# RETAIL_SALES
# Returns the monthly Advance Retail Sales: Retail Trade data of the United States.

import requests

url = 'https://www.alphavantage.co/query?function=RETAIL_SALES&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

print(json.dumps(data, indent=2))
```

```
In [ ]: # DURABLES
        # Returns the monthly manufacturers' new orders of durable goods in the United States.

        import requests

        url = 'https://www.alphavantage.co/query?function=DURABLES&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        print(json.dumps(data, indent=2))
```

```
In [ ]: # UNEMPLOYMENT
        # Returns the monthly unemployment data of the United States. The unemployment rate represents the number of unemployed as a percentage of the labor force. Labor force data are restricted to people 16 years of age an

        import requests

        url = 'https://www.alphavantage.co/query?function=UNEMPLOYMENT&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        print(json.dumps(data, indent=2))
```

```
In [ ]: # NONFARM_PAYROLL
        # Returns the monthly US All Employees: Total Nonfarm (commonly known as Total Nonfarm Payroll), a measure of the number of U.S. workers in the economy that excludes proprietors, private household employees, unpaid v

        import requests

        url = 'https://www.alphavantage.co/query?function=NONFARM_PAYROLL&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        print(json.dumps(data, indent=2))
```

```
In [ ]: # WMA
        # Returns the weighted moving average (WMA) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=WMA&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: WMA"]

        for x in timeSeries:
            wma_day = x
            wma_name = data["Meta Data"]["1: Symbol"]
            wma_value = timeSeries[x]["WMA"]

            try:
                query = "INSERT INTO wma(wma_day, wma_name, wma_value)" \
                        "VALUES(%s,%s,%s)"
                args = (wma_day, wma_name, wma_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```python
# DEMA
# Returns the double exponential moving average (DEMA) values.

import requests

url = 'https://www.alphavantage.co/query?function=DEMA&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: DEMA"]

for x in timeSeries:
    dema_day = x
    dema_name = data["Meta Data"]["1: Symbol"]
    dema_value = timeSeries[x]["DEMA"]

    try:
        query = "INSERT INTO dema(dema_day, dema_name, dema_value)" \
                "VALUES(%s,%s,%s)"
        args = (dema_day, dema_name, dema_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# TEMA
# Returns the triple exponential moving average (TEMA) values.

import requests

url = 'https://www.alphavantage.co/query?function=TEMA&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: TEMA"]

for x in timeSeries:
    tema_day = x
    tema_name = data["Meta Data"]["1: Symbol"]
    tema_value = timeSeries[x]["TEMA"]

    try:
        query = "INSERT INTO tema(tema_day, tema_name, tema_value)" \
                "VALUES(%s,%s,%s)"
        args = (tema_day, tema_name, tema_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
In [ ]:  # TRIMA

         # Returns the triangular moving average (TRIMA) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=TRIMA&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: TRIMA"]

         for x in timeSeries:
             trima_day = x
             trima_name = data["Meta Data"]["1: Symbol"]
             trima_value = timeSeries[x]["TRIMA"]

             try:
                 query = "INSERT INTO trima(trima_day, trima_name, trima_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (trima_day, trima_name, trima_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))


In [ ]:  # KAMA

         # Returns the Kaufman adaptive moving average (KAMA) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=KAMA&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: KAMA"]

         for x in timeSeries:
             kama_day = x
             kama_name = data["Meta Data"]["1: Symbol"]
             kama_value = timeSeries[x]["KAMA"]

             try:
                 query = "INSERT INTO kama(kama_day, kama_name, kama_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (kama_day, kama_name, kama_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')


         # print(json.dumps(data, indent=2))
```

```
In [ ]: # MAMA

        # Returns the MESA adaptive moving average (MAMA) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=MAMA&symbol=AAPL&interval=daily&series_type=close&fastlimit=0.02&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: MAMA"]

        for x in timeSeries:
            mama_day = x
            mama_name = data["Meta Data"]["1: Symbol"]
            mama_value = timeSeries[x]["MAMA"]

            try:
                query = "INSERT INTO mama(mama_day, mama_name, mama_value)" \
                        "VALUES(%s,%s,%s)"
                args = (mama_day, mama_name, mama_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')


        # print(json.dumps(data, indent=2))
```

```
In [ ]: # T3

        # Returns the triple exponential moving average (T3) values

        import requests

        url = 'https://www.alphavantage.co/query?function=T3&symbol=AAPL&interval=weekly&time_period=10&series_type=open&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: T3"]

        for x in timeSeries:
            t3_day = x
            t3_name = data["Meta Data"]["1: Symbol"]
            t3_value = timeSeries[x]["T3"]

            try:
                query = "INSERT INTO t3(t3_day, t3_name, t3_value)" \
                        "VALUES(%s,%s,%s)"
                args = (t3_day, t3_name, t3_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```
In [ ]:  # MACDEXT

         # Returns the moving average convergence / divergence values with controllable moving average type.

         import requests

         url = 'https://www.alphavantage.co/query?function=MACDEXT&symbol=AAPL&interval=daily&series_type=open&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: MACDEXT"]

         for x in timeSeries:
             MACD_day = x
             MACD_name = data["Meta Data"]["1: Symbol"]
             MACD_Signal = timeSeries[x]["MACD_Signal"]
             MACD_Hist = timeSeries[x]["MACD_Hist"]
             MACD_value = timeSeries[x]["MACD"]

             try:
                 query = "INSERT INTO macd(MACD_day, MACD_name, MACD_Signal, MACD_Hist, MACD_value)" \
                         "VALUES(%s,%s,%s,%s,%s)"
                 args = (MACD_day, MACD_name, MACD_Signal, MACD_Hist, MACD_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))

In [ ]:  # STOCHF

         # Returns the stochastic fast (STOCHF) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=STOCHF&symbol=AAPL&interval=daily&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: STOCHF"]

         for x in timeSeries:
             STOCHF_day = x
             STOCHF_name = data["Meta Data"]["1: Symbol"]
             FastD = timeSeries[x]["FastD"]
             FastK = timeSeries[x]["FastK"]

             try:
                 query = "INSERT INTO stochf(STOCHF_day, STOCHF_name, FastD, FastK)" \
                         "VALUES(%s,%s,%s,%s)"
                 args = (STOCHF_day, STOCHF_name, FastD, FastK)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```
In [ ]:   # STOCHRSI

          # Returns the stochastic relative strength index (STOCHRSI) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=STOCHRSI&symbol=AAPL&interval=daily&time_period=10&series_type=close&fastkperiod=6&fastdmatype=1&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: STOCHRSI"]

          for x in timeSeries:
              STOCHRSI_day = x
              STOCHRSI_name = data["Meta Data"]["1: Symbol"]
              FastD = timeSeries[x]["FastD"]
              FastK = timeSeries[x]["FastK"]

              try:
                  query = "INSERT INTO stochrsi(STOCHRSI_day, STOCHRSI_name, FastD, FastK)" \
                          "VALUES(%s,%s,%s,%s)"
                  args = (STOCHRSI_day, STOCHRSI_name, FastD, FastK)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')


          # print(json.dumps(data, indent=2))

In [ ]:   # WILLR

          # Returns the Williams' %R (WILLR) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=WILLR&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: WILLR"]

          for x in timeSeries:
              WILLR_day = x
              WILLR_name = data["Meta Data"]["1: Symbol"]
              WILLR_value = timeSeries[x]["WILLR"]

              try:
                  query = "INSERT INTO willr(WILLR_day, WILLR_name, WILLR_value)" \
                          "VALUES(%s,%s,%s)"
                  args = (WILLR_day, WILLR_name, WILLR_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))
```

```
# ADXR

# Returns the average directional movement index rating (ADXR) values.

import requests

url = 'https://www.alphavantage.co/query?function=ADXR&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: ADXR"]

for x in timeSeries:
    ADXR_day = x
    ADXR_name = data["Meta Data"]["1: Symbol"]
    ADXR_value = timeSeries[x]["ADXR"]

    try:
        query = "INSERT INTO adxr(ADXR_day, ADXR_name, ADXR_value)" \
                "VALUES(%s,%s,%s)"
        args = (ADXR_day, ADXR_name, ADXR_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
# APO

# Returns the absolute price oscillator (APO) values.

import requests

url = 'https://www.alphavantage.co/query?function=APO&symbol=AAPL&interval=daily&series_type=close&fastperiod=10&matype=1&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: APO"]

for x in timeSeries:
    APO_day = x
    APO_name = data["Meta Data"]["1: Symbol"]
    APO_value = timeSeries[x]["APO"]

    try:
        query = "INSERT INTO apo(APO_day, APO_name, APO_value)" \
                "VALUES(%s,%s,%s)"
        args = (APO_day, APO_name, APO_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
In [ ]:   # PPO

          # Returns the percentage price oscillator (PPO) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=PPO&symbol=IBM&interval=daily&series_type=close&fastperiod=10&matype=1&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: PPO"]

          for x in timeSeries:
              PPO_day = x
              PPO_name = data["Meta Data"]["1: Symbol"]
              PPO_value = timeSeries[x]["PPO"]

              try:
                  query = "INSERT INTO ppo(PPO_day, PPO_name, PPO_value)" \
                          "VALUES(%s,%s,%s)"
                  args = (PPO_day, PPO_name, PPO_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))

In [ ]:   # MOM

          # Returns the momentum (MOM) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=MOM&symbol=AAPL&interval=daily&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: MOM"]

          for x in timeSeries:
              MOM_day = x
              MOM_name = data["Meta Data"]["1: Symbol"]
              MOM_value = timeSeries[x]["MOM"]

              try:
                  query = "INSERT INTO mom(MOM_day, MOM_name, MOM_value)" \
                          "VALUES(%s,%s,%s)"
                  args = (MOM_day, MOM_name, MOM_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))
```

```python
In [ ]:  # BOP

         # Returns the balance of power (BOP) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=BOP&symbol=IBM&interval=daily&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: BOP"]

         for x in timeSeries:
             BOP_day = x
             BOP_name = data["Meta Data"]["1: Symbol"]
             BOP_value = timeSeries[x]["BOP"]

             try:
                 query = "INSERT INTO bop(BOP_day, BOP_name, BOP_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (BOP_day, BOP_name, BOP_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')


         # print(json.dumps(data, indent=2))
```

```python
In [ ]:  # CMO

         # Returns the Chande momentum oscillator (CMO) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=CMO&symbol=IBM&interval=weekly&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: CMO"]

         for x in timeSeries:
             CMO_day = x
             CMO_name = data["Meta Data"]["1: Symbol"]
             CMO_value = timeSeries[x]["CMO"]

             try:
                 query = "INSERT INTO cmo(CMO_day, CMO_name, CMO_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (CMO_day, CMO_name, CMO_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')


         # print(json.dumps(data, indent=2))
```

```python
# ROC

# Returns the rate of change (ROC) values.

import requests

url = 'https://www.alphavantage.co/query?function=ROC&symbol=IBM&interval=weekly&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: ROC"]

for x in timeSeries:
    ROC_day = x
    ROC_name = data["Meta Data"]["1: Symbol"]
    ROC_value = timeSeries[x]["ROC"]

    try:
        query = "INSERT INTO roc(ROC_day, ROC_name, ROC_value)" \
                "VALUES(%s,%s,%s)"
        args = (ROC_day, ROC_name, ROC_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# ROCR

# Returns the rate of change ratio (ROCR) values.

import requests

url = 'https://www.alphavantage.co/query?function=ROCR&symbol=IBM&interval=daily&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: ROCR"]

for x in timeSeries:
    ROCR_day = x
    ROCR_name = data["Meta Data"]["1: Symbol"]
    ROCR_value = timeSeries[x]["ROCR"]

    try:
        query = "INSERT INTO rocr(ROCR_day, ROCR_name, ROCR_value)" \
                "VALUES(%s,%s,%s)"
        args = (ROCR_day, ROCR_name, ROCR_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# AROON

# This API returns the Aroon (AROON) values.

import requests

url = 'https://www.alphavantage.co/query?function=AROON&symbol=AAPL&interval=daily&time_period=14&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: AROON"]

for x in timeSeries:
    AROON_day = x
    AROON_name = data["Meta Data"]["1: Symbol"]
    Aroon_Up = timeSeries[x]["Aroon Up"]
    Aroon_Down = timeSeries[x]["Aroon Down"]

    try:
        query = "INSERT INTO aroon(AROON_day, AROON_name, Aroon_Up, Aroon_Down)" \
                "VALUES(%s,%s,%s,%s)"
        args = (AROON_day, AROON_name, Aroon_Up, Aroon_Down)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')


# print(json.dumps(data, indent=2))
```

```python
# AROONOSC

# Returns the Aroon oscillator (AROONOSC) values.

import requests

url = 'https://www.alphavantage.co/query?function=AROONOSC&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: AROONOSC"]

for x in timeSeries:
    AROONOSC_day = x
    AROONOSC_name = data["Meta Data"]["1: Symbol"]
    AROONOSC_value = timeSeries[x]["AROONOSC"]

    try:
        query = "INSERT INTO aroonosc(AROONOSC_day, AROONOSC_name, AROONOSC_value)" \
                "VALUES(%s,%s,%s)"
        args = (AROONOSC_day, AROONOSC_name, AROONOSC_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
# MFI

# Returns the money flow index (MFI) values.

import requests

url = 'https://www.alphavantage.co/query?function=MFI&symbol=AAPL&interval=weekly&time_period=10&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: MFI"]

for x in timeSeries:
    MFI_day = x
    MFI_name = data["Meta Data"]["1: Symbol"]
    MFI_value = timeSeries[x]["MFI"]

    try:
        query = "INSERT INTO mfi(MFI_day, MFI_name, MFI_value)" \
                "VALUES(%s,%s,%s)"
        args = (MFI_day, MFI_name, MFI_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
# TRIX

# Returns the 1-day rate of change of a triple smooth exponential moving average (TRIX) values.

import requests

url = 'https://www.alphavantage.co/query?function=TRIX&symbol=AAPL&interval=daily&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: TRIX"]

for x in timeSeries:
    TRIX_day = x
    TRIX_name = data["Meta Data"]["1: Symbol"]
    TRIX_value = timeSeries[x]["TRIX"]

    try:
        query = "INSERT INTO trix(TRIX_day, TRIX_name, TRIX_value)" \
                "VALUES(%s,%s,%s)"
        args = (TRIX_day, TRIX_name, TRIX_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# ULTOSC

# Returns the ultimate oscillator (ULTOSC) values

import requests

url = 'https://www.alphavantage.co/query?function=ULTOSC&symbol=IBM&interval=daily&timeperiod1=8&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: ULTOSC"]

for x in timeSeries:
    ULTOSC_day = x
    ULTOSC_name = data["Meta Data"]["1: Symbol"]
    ULTOSC_value = timeSeries[x]["ULTOSC"]

    try:
        query = "INSERT INTO ultosc(ULTOSC_day, ULTOSC_name, ULTOSC_value)" \
                "VALUES(%s,%s,%s)"
        args = (ULTOSC_day, ULTOSC_name, ULTOSC_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# DX

# This API returns the directional movement index (DX) values.

import requests

url = 'https://www.alphavantage.co/query?function=DX&symbol=IBM&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: DX"]

for x in timeSeries:
    DX_day = x
    DX_name = data["Meta Data"]["1: Symbol"]
    DX_value = timeSeries[x]["DX"]

    try:
        query = "INSERT INTO dx(DX_day, DX_name, DX_value)" \
                "VALUES(%s,%s,%s)"
        args = (DX_day, DX_name, DX_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
In [ ]: # MINUS_DI

        # This API returns the minus directional indicator (MINUS_DI) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=MINUS_DI&symbol=AAPL&interval=weekly&time_period=10&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: MINUS_DI"]

        for x in timeSeries:
            MINUS_DI_day = x
            MINUS_DI_name = data["Meta Data"]["1: Symbol"]
            MINUS_DI_value = timeSeries[x]["MINUS_DI"]

            try:
                query = "INSERT INTO minusdi(MINUS_DI_day, MINUS_DI_name, MINUS_DI_value)" \
                        "VALUES(%s,%s,%s)"
                args = (MINUS_DI_day, MINUS_DI_name, MINUS_DI_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```
In [ ]: # PLUS_DI

        # This API returns the plus directional indicator (PLUS_DI) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=PLUS_DI&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: PLUS_DI"]

        for x in timeSeries:
            PLUS_DI_day = x
            PLUS_DI_name = data["Meta Data"]["1: Symbol"]
            PLUS_DI_value = timeSeries[x]["PLUS_DI"]

            try:
                query = "INSERT INTO plusdi(PLUS_DI_day, PLUS_DI_name, PLUS_DI_value)" \
                        "VALUES(%s,%s,%s)"
                args = (PLUS_DI_day, PLUS_DI_name, PLUS_DI_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```
In [ ]:  # MINUS_DM

         # This API returns the minus directional movement (MINUS_DM) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=MINUS_DM&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: MINUS_DM"]

         for x in timeSeries:
             MINUS_DM_day = x
             MINUS_DM_name = data["Meta Data"]["1: Symbol"]
             MINUS_DM_value = timeSeries[x]["MINUS_DM"]

             try:
                 query = "INSERT INTO minusdm(MINUS_DM_day, MINUS_DM_name, MINUS_DM_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (MINUS_DM_day, MINUS_DM_name, MINUS_DM_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```
In [ ]:  # PLUS_DM

         # This API returns the minus directional movement (MINUS_DM) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=PLUS_DM&symbol=AAPL&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: PLUS_DM"]

         for x in timeSeries:
             PLUS_DM_day = x
             PLUS_DM_name = data["Meta Data"]["1: Symbol"]
             PLUS_DM_value = timeSeries[x]["PLUS_DM"]

             try:
                 query = "INSERT INTO plusdm(PLUS_DM_day, PLUS_DM_name, PLUS_DM_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (PLUS_DM_day, PLUS_DM_name, PLUS_DM_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```python
# MIDPOINT

# This API returns the midpoint (MIDPOINT) values. MIDPOINT = (highest value + lowest value)/2.

import requests

url = 'https://www.alphavantage.co/query?function=MIDPOINT&symbol=AAPL&interval=daily&time_period=10&series_type=close&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: MIDPOINT"]

for x in timeSeries:
    MIDPOINT_day = x
    MIDPOINT_name = data["Meta Data"]["1: Symbol"]
    MIDPOINT_value = timeSeries[x]["MIDPOINT"]

    try:
        query = "INSERT INTO midpoint(MIDPOINT_day, MIDPOINT_name, MIDPOINT_value)" \
                "VALUES(%s,%s,%s)"
        args = (MIDPOINT_day, MIDPOINT_name, MIDPOINT_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# MIDPRICE

# This API returns the midpoint price (MIDPRICE) values. MIDPRICE = (highest high + lowest low)/2.

import requests

url = 'https://www.alphavantage.co/query?function=MIDPRICE&symbol=IBM&interval=daily&time_period=10&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: MIDPRICE"]

for x in timeSeries:
    MIDPRICE_day = x
    MIDPRICE_name = data["Meta Data"]["1: Symbol"]
    MIDPRICE_value = timeSeries[x]["MIDPRICE"]

    try:
        query = "INSERT INTO midprice(MIDPRICE_day, MIDPRICE_name, MIDPRICE_value)" \
                "VALUES(%s,%s,%s)"
        args = (MIDPRICE_day, MIDPRICE_name, MIDPRICE_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
In [ ]:  # SAR

         # This API returns the parabolic SAR (SAR) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=SAR&symbol=AAPL&interval=weekly&acceleration=0.05&maximum=0.25&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: SAR"]

         for x in timeSeries:
             SAR_day = x
             SAR_name = data["Meta Data"]["1: Symbol"]
             SAR_value = timeSeries[x]["SAR"]

             try:
                 query = "INSERT INTO sar(SAR_day, SAR_name, SAR_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (SAR_day, SAR_name, SAR_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))


In [ ]:  # TRANGE

         # This API returns the true range (TRANGE) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=TRANGE&symbol=AAPL&interval=daily&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: TRANGE"]

         for x in timeSeries:
             TRANGE_day = x
             TRANGE_name = data["Meta Data"]["1: Symbol"]
             TRANGE_value = timeSeries[x]["TRANGE"]

             try:
                 query = "INSERT INTO trange(TRANGE_day, TRANGE_name, TRANGE_value)" \
                         "VALUES(%s,%s,%s)"
                 args = (TRANGE_day, TRANGE_name, TRANGE_value)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```python
# ATR

# This API returns the average true range (ATR) values.

import requests

url = 'https://www.alphavantage.co/query?function=ATR&symbol=AAPL&interval=daily&time_period=14&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: ATR"]

for x in timeSeries:
    ATR_day = x
    ATR_name = data["Meta Data"]["1: Symbol"]
    ATR_value = timeSeries[x]["ATR"]

    try:
        query = "INSERT INTO atr(ATR_day, ATR_name, ATR_value)" \
                "VALUES(%s,%s,%s)"
        args = (ATR_day, ATR_name, ATR_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```python
# NATR

# This API returns the average true range (NATR) values.

import requests

url = 'https://www.alphavantage.co/query?function=NATR&symbol=AAPL&interval=weekly&time_period=14&apikey=R8QBN54GF80WJUT6'
r = requests.get(url)
data = r.json()

import mysql.connector
import os

try:
    conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
    print("Connection established to", conn.database)
except:
    print("Error connecting to ", 'glance_at_finance')

timeSeries = data["Technical Analysis: NATR"]

for x in timeSeries:
    NATR_day = x
    NATR_name = data["Meta Data"]["1: Symbol"]
    NATR_value = timeSeries[x]["NATR"]

    try:
        query = "INSERT INTO Natr(NATR_day, NATR_name, NATR_value)" \
                "VALUES(%s,%s,%s)"
        args = (NATR_day, NATR_name, NATR_value)
        conn.cursor().execute(query, args)
        conn.commit()
    except:
        print("Error during insertion")

# closing the connection
try:
    conn.close()
except:
    print('No Connection found')

# print(json.dumps(data, indent=2))
```

```
In [ ]: # AD

        # This API returns the Chaikin A/D line (AD) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=AD&symbol=IBM&interval=daily&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: Chaikin A/D"]

        for x in timeSeries:
            AD_day = x
            AD_name = data["Meta Data"]["1: Symbol"]
            AD_value = timeSeries[x]["Chaikin A/D"]

            try:
                query = "INSERT INTO ad(AD_day, AD_name, AD_value)" \
                        "VALUES(%s,%s,%s)"
                args = (AD_day, AD_name, AD_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```
In [ ]: # ADOSC

        # This API returns the Chaikin A/D oscillator (ADOSC) values.

        import requests

        url = 'https://www.alphavantage.co/query?function=ADOSC&symbol=AAPL&interval=daily&fastperiod=5&apikey=R8QBN54GF80WJUT6'
        r = requests.get(url)
        data = r.json()

        import mysql.connector
        import os

        try:
            conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
            print("Connection established to", conn.database)
        except:
            print("Error connecting to ", 'glance_at_finance')

        timeSeries = data["Technical Analysis: ADOSC"]

        for x in timeSeries:
            ADOSC_day = x
            ADOSC_name = data["Meta Data"]["1: Symbol"]
            ADOSC_value = timeSeries[x]["ADOSC"]

            try:
                query = "INSERT INTO adosc(ADOSC_day, ADOSC_name, ADOSC_value)" \
                        "VALUES(%s,%s,%s)"
                args = (ADOSC_day, ADOSC_name, ADOSC_value)
                conn.cursor().execute(query, args)
                conn.commit()
            except:
                print("Error during insertion")

        # closing the connection
        try:
            conn.close()
        except:
            print('No Connection found')

        # print(json.dumps(data, indent=2))
```

```
In [ ]:   # OBV

          # This API returns the on balance volume (OBV) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=OBV&symbol=IBM&interval=weekly&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: OBV"]

          for x in timeSeries:
              OBV_day = x
              OBV_name = data["Meta Data"]["1: Symbol"]
              OBV_value = timeSeries[x]["OBV"]

              try:
                  query = "INSERT INTO obv(OBV_day, OBV_name, OBV_value)" \
                          "VALUES(%s,%s,%s)"
                  args = (OBV_day, OBV_name, OBV_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))
```

```
In [ ]:   # HT_TRENDLINE

          # This API returns the Hilbert transform, instantaneous trendline (HT_TRENDLINE) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=HT_TRENDLINE&symbol=AAPL&interval=daily&series_type=close&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: HT_TRENDLINE"]

          for x in timeSeries:
              HT_TRENDLINE_day = x
              HT_TRENDLINE_name = data["Meta Data"]["1: Symbol"]
              HT_TRENDLINE_value = timeSeries[x]["HT_TRENDLINE"]

              try:
                  query = "INSERT INTO httrendline(HT_TRENDLINE_day, HT_TRENDLINE_name, HT_TRENDLINE_value)" \
                          "VALUES(%s,%s,%s)"
                  args = (HT_TRENDLINE_day, HT_TRENDLINE_name, HT_TRENDLINE_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))
```

```
In [ ]:  # HT_SINE

         # This API returns the Hilbert transform, instantaneous trendline (HT_TRENDLINE) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=HT_SINE&symbol=AAPL&interval=daily&series_type=close&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: HT_SINE"]

         for x in timeSeries:
             HT_SINE_day = x
             HT_SINE_name = data["Meta Data"]["1: Symbol"]
             LEAD_SINE = timeSeries[x]["LEAD SINE"]
             SINE = timeSeries[x]["SINE"]

             try:
                 query = "INSERT INTO htsine(HT_SINE_day, HT_SINE_name, LEAD_SINE, SINE)" \
                         "VALUES(%s,%s,%s,%s)"
                 args = (HT_SINE_day, HT_SINE_name, LEAD_SINE, SINE)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```
In [ ]:  # HT_TRENDMODE

         # This API returns the Hilbert transform, trend vs cycle mode (HT_TRENDMODE) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=HT_TRENDMODE&symbol=AAPL&interval=weekly&series_type=close&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: HT_TRENDMODE"]

         for x in timeSeries:
             HT_TRENDMODE_day = x
             HT_TRENDMODE_name = data["Meta Data"]["1: Symbol"]
             HT_TRENDMODE_VALUE = timeSeries[x]["TRENDMODE"]

             try:
                 query = "INSERT INTO httrendmode(HT_TRENDMODE_day, HT_TRENDMODE_name, HT_TRENDMODE_VALUE)" \
                         "VALUES(%s,%s,%s)"
                 args = (HT_TRENDMODE_day, HT_TRENDMODE_name, HT_TRENDMODE_VALUE)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```
In [ ]:  # HT_DCPERIOD

         # This API returns the Hilbert transform, dominant cycle period (HT_DCPERIOD) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=HT_DCPERIOD&symbol=AAPL&interval=daily&series_type=close&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: HT_DCPERIOD"]

         for x in timeSeries:
             HT_DCPERIOD_day = x
             HT_DCPERIOD_name = data["Meta Data"]["1: Symbol"]
             HT_DCPERIOD_VALUE = timeSeries[x]["DCPERIOD"]

             try:
                 query = "INSERT INTO htdcperiod(HT_DCPERIOD_day, HT_DCPERIOD_name, HT_DCPERIOD_VALUE)" \
                         "VALUES(%s,%s,%s)"
                 args = (HT_DCPERIOD_day, HT_DCPERIOD_name, HT_DCPERIOD_VALUE)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))


In [ ]:  # HT_PHASE

         # This API returns the Hilbert transform, dominant cycle phase (HT_DCPHASE) values.

         import requests

         url = 'https://www.alphavantage.co/query?function=HT_DCPHASE&symbol=AAPL&interval=daily&series_type=close&apikey=R8QBN54GF80WJUT6'
         r = requests.get(url)
         data = r.json()

         import mysql.connector
         import os

         try:
             conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
             print("Connection established to", conn.database)
         except:
             print("Error connecting to ", 'glance_at_finance')

         timeSeries = data["Technical Analysis: HT_DCPHASE"]

         for x in timeSeries:
             HT_DCPHASE_day = x
             HT_DCPHASE_name = data["Meta Data"]["1: Symbol"]
             HT_DCPHASE_VALUE = timeSeries[x]["HT_DCPHASE"]

             try:
                 query = "INSERT INTO htdcphase(HT_DCPHASE_day, HT_DCPHASE_name, HT_DCPHASE_VALUE)" \
                         "VALUES(%s,%s,%s)"
                 args = (HT_DCPHASE_day, HT_DCPHASE_name, HT_DCPHASE_VALUE)
                 conn.cursor().execute(query, args)
                 conn.commit()
             except:
                 print("Error during insertion")

         # closing the connection
         try:
             conn.close()
         except:
             print('No Connection found')

         # print(json.dumps(data, indent=2))
```

```
In [ ]:   # HT_PHASOR

          # This API returns the Hilbert transform, phasor components (HT_PHASOR) values.

          import requests

          url = 'https://www.alphavantage.co/query?function=HT_PHASOR&symbol=AAPL&interval=weekly&series_type=close&apikey=R8QBN54GF80WJUT6'
          r = requests.get(url)
          data = r.json()

          import mysql.connector
          import os

          try:
              conn = mysql.connector.connect(user = 'root', password = 'bakugan56', host = '127.0.0.1', database = 'glance_at_finance')
              print("Connection established to", conn.database)
          except:
              print("Error connecting to ", 'glance_at_finance')

          timeSeries = data["Technical Analysis: HT_PHASOR"]

          for x in timeSeries:
              HT_PHASOR_day = x
              HT_PHASOR_name = data["Meta Data"]["1: Symbol"]
              QUADRATURE = timeSeries[x]["QUADRATURE"]
              PHASE_value = timeSeries[x]["PHASE"]

              try:
                  query = "INSERT INTO htphasor(HT_PHASOR_day, HT_PHASOR_name, QUADRATURE, PHASE_value)" \
                          "VALUES(%s,%s,%s,%s)"
                  args = (HT_PHASOR_day, HT_PHASOR_name, QUADRATURE, PHASE_value)
                  conn.cursor().execute(query, args)
                  conn.commit()
              except:
                  print("Error during insertion")

          # closing the connection
          try:
              conn.close()
          except:
              print('No Connection found')

          # print(json.dumps(data, indent=2))

In [ ]:
```