FINAL PROJECT
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,
```

```
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,
```

```
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)
);
```

```
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)
```

```
);

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(
```

```
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,
```

```
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(
```

```
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)
);
```

```
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(
```

```
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)
);

select * from daily_adjusted;
select * from weekly;
select * from monthly;
select * from digital_currency_exchange;
select * from digital_currency_weekly;
select * from digital_currency_monthly;
select * from wma;
select * from dema;
select * from tema;
select * from trima;
select * from kama;
select * from mama;
select * from t3;
select * from macd;
select * from stochf;
select * from stochrsi;
select * from willr;
select * from adxr;
select * from apo;
select * from ppo;
select * from mom;
select * from bop;
select * from cmo;
select * from roc;
select * from rocr;
select * from aroon;
```

```
select * from aroonosc;
select * from mfi;
select * from trix;
select * from ultosc;
select * from dx;
select * from minusdi;
select * from plusdi;
select * from minusdm;
select * from plusdm;
select * from midpoint;
select * from midprice;
select * from sar;
select * from trange;
select * from atr;
select * from natr;
select * from ad;
select * from adosc;
select * from obv;
select * from httrendline;
select * from htsine;
select * from httrendmode;
select * from htdcperiod;
select * from htdcphase;
select * from htphasor;
```

ER DIAGRAM:


roc
ROC_name VARCHAR(100)
ROC_day VARCHAR(20)
ROC_value FLOAT
mom
MOM_name VARCHAR(100)
MOM_day VARCHAR(20)
MOM_value FLOAT
di_analysis_view
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
tema
tema_name VARCHAR(100)
tema_day VARCHAR(20)
tema_value FLOAT
ad
AD_name VARCHAR(100)
AD_day VARCHAR(20)
AD_value FLOAT
earnings
stock_name VARCHAR(100)
fiscalDateEnding VARCHAR(20)
reportedEPS 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
adosc
ADOSC_name VARCHAR(100)
ADOSC_day VARCHAR(20)
ADOSC_value FLOAT
di_analysis
natr
NATR_name VARCHAR(100)
NATR_day VARCHAR(20)
NATR_value 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
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
plusdm
PLUS_DM_name VARCHAR(100)
PLUS_DM_day VARCHAR(20)
PLUS_DM_value FLOAT
atr
ATR_name VARCHAR(100)
ATR_day VARCHAR(20)
ATR_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
trix
TRIX_name VARCHAR(100)
TRIX_day VARCHAR(20)
TRIX_value FLOAT
macd
macd_name VARCHAR(100)
macd_day VARCHAR(20)
MACD_Signal FLOAT
MACD_Hist FLOAT
MACD_value FLOAT
rocr
ROCR_name VARCHAR(100)
ROCR_day VARCHAR(20)
ROCR_value FLOAT
sar
SAR_name VARCHAR(100)
SAR_day VARCHAR(20)
SAR_value FLOAT
kama
kama_name VARCHAR(100)
kama_day VARCHAR(20)
kama_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
plusdi
PLUS_DI_name VARCHAR(100)
PLUS_DI_day VARCHAR(20)
PLUS_DI_value FLOAT
htdcphase
HT_DCPHASE_name VARCHAR(100)
HT_DCPHASE_day VARCHAR(20)
HT_DCPHASE_VALUE FLOAT
aroon_view
htsine
HT_SINE_name VARCHAR(100)
HT_SINE_day VARCHAR(20)
LEAD_SINE FLOAT
SINE FLOAT
ma_view
htdcperiod
HT_DCPERIOD_name VARCHAR(100)
HT_DCPERIOD_day VARCHAR(20)
HT_DCPERIOD_VALUE FLOAT
stochrsi
STOCHRSI_name VARCHAR(100)
STOCHRSI_day VARCHAR(20)
FastD FLOAT
FastK 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
apo
APO_name VARCHAR(100)
APO_day VARCHAR(20)
APO_value FLOAT
midprice
MIDPRICE_name VARCHAR(100)
MIDPRICE_day VARCHAR(20)
MIDPRICE_value FLOAT
willr
WILLR_name VARCHAR(100)
WILLR_day VARCHAR(20)
WILLR_value FLOAT
cmo
CMO_name VARCHAR(100)
CMO_day VARCHAR(20)
CMO_value FLOAT
httrendmode
HT_TRENDMODE_name VARCHAR(100)
HT_TRENDMODE_day VARCHAR(20)
HT_TRENDMODE_VALUE FLOAT
stochf
STOCHF_name VARCHAR(100)
STOCHF_day VARCHAR(20)
FastD FLOAT
FastK FLOAT
ultosc
ULTOSC_name VARCHAR(100)
ULTOSC_day VARCHAR(20)
ULTOSC_value FLOAT
t3
t3_name VARCHAR(100)
t3_day VARCHAR(20)
t3_value FLOAT
adxr
ADXR_name VARCHAR(100)
ADXR_day VARCHAR(20)
ADXR_value FLOAT
wma
wma_name VARCHAR(100)
wma_day VARCHAR(20)
wma_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
digital_currency_exchange
From_Currency VARCHAR(20)
To_Currency VARCHAR(20)
Exchange_Rate FLOAT
obv
OBV_name VARCHAR(100)
OBV_day VARCHAR(20)
OBV_value FLOAT
fxexchange
From_Currency VARCHAR(100)
To_Currency VARCHAR(100)
Exchange_Rate FLOAT
mfi
MFI_name VARCHAR(100)
MFI_day VARCHAR(20)
MFI_value FLOAT
dx
DX_name VARCHAR(100)
DX_day VARCHAR(20)
DX_value FLOAT
htphasor
HT_PHASOR_name VARCHAR(100)
HT_PHASOR_day VARCHAR(20)
QUADRATURE FLOAT
PHASE_value FLOAT
ppo
PPO_name VARCHAR(100)
PPO_day VARCHAR(20)
PPO_value FLOAT
aroonosc
AROONOSC_name VARCHAR(100)
AROONOSC_day VARCHAR(20)
AROONOSC_value FLOAT
bop
BOP_name VARCHAR(100)
BOP_day VARCHAR(20)
BOP_value FLOAT
aroon
AROON_name VARCHAR(100)
AROON_day VARCHAR(20)
Aroon_Up FLOAT
Aroon_Down FLOAT
trima
trima_name VARCHAR(100)
trima_day VARCHAR(20)
trima_value FLOAT
ht_view
midpoint
MIDPOINT_name VARCHAR(100)
MIDPOINT_day VARCHAR(20)
MIDPOINT_value FLOAT
httrendline
HT_TRENDLINE_name VARCHAR(100)
HT_TRENDLINE_day VARCHAR(20)
HT_TRENDLINE_value 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
minusdm
MINUS_DM_name VARCHAR(100)
MINUS_DM_day VARCHAR(20)
MINUS_DM_value FLOAT
minusdi
MINUS_DI_name VARCHAR(100)
MINUS_DI_day VARCHAR(20)
MINUS_DI_value FLOAT
mama
mama_name VARCHAR(100)
mama_day VARCHAR(20)
mama_value FLOAT

NORMALIZATION:

All the data when extracted was sent to their own unique tables with functionality already set to satisfy the Normal forms as the data needed to be pulled live for the final trading application. Keeping that in mind, the initial extraction was done in a way that data didn't have any redundancy, no duplicates in the same table and only took data values of one type or stock at a time to make sure there were no errors due to confusion of which function to pull.
Following are the Tables and Data Values for tables:

stock_name	stock_day	stock_open	stock_high	stock_low	stock_close	stock_volume
AAPL	1999-12-31	101	118	91.06	102.81	84091200
AAPL	2000-01-31	104.87	121.5	86.5	103.75	112100000
AAPL	2000-02-29	104	119.94	97	114.62	65355200
AAPL	2000-03-31	118.56	150.38	114	135.81	77663900
AAPL	2000-04-28	135.5	139.5	104.87	124.06	77342900
AAPL	2000-05-31	124.87	126.25	81.75	84	87569200
AAPL	2000-06-30	81.75	103.94	50.31	52.38	89106200
AAPL	2000-07-31	52.13	60.63	46.88	50.81	102621000
AAPL	2000-08-31	50.31	61.5	44.25	60.94	100644000
AAPL	2000-09-29	61.31	64.12	25.37	25.75	259231000
AAPL	2000-10-31	26.69	26.75	17.5	19.56	391175000
AAPL	2000-11-30	19.44	23	16.12	16.5	151578000
AAPL	2000-12-29	17	17.5	13.63	14.88	158196000
AAPL	2001-01-31	14.88	22.5	14.44	21.62	244812000
AAPL	2001-02-28	20.69	21.94	18	18.25	125424000
AAPL	2001-03-30	17.81	23.75	17.19	22.07	192840000
AAPL	2001-04-30	22.09	27.12	18.75	25.49	199268000
AAPL	2001-05-31	25.41	26.7	19.3	19.95	133365000
AAPL	2001-06-29	20.13	25.1	19.35	23.25	136398000
AAPL	2001-07-31	23.64	25.22	17.85	18.79	154555000
AAPL	2001-08-31	19.01	19.9	17.28	18.55	91595700
AAPL	2001-09-28	18.5	19.08	14.68	15.51	98777200
AAPL	2001-10-31	15.49	19.42	14.83	17.56	134719000
AAPL	2001-11-30	17.65	21.55	17.25	21.3	95888300
AAPL	2001-12-31	21.06	24.03	20.09	21.9	82675700
AAPL	2002-01-31	22.05	24.73	20.26	24.72	152060000

	WILLR_name	WILLR_day	WILLR_value
▶	AAPL	1999-11-12	-34.8188
	AAPL	1999-11-15	-40.5975
	AAPL	1999-11-16	-38.2233
	AAPL	1999-11-17	-43.7171
	AAPL	1999-11-18	-59.0677
	AAPL	1999-11-19	-48.1785
	AAPL	1999-11-22	-55.9621
	AAPL	1999-11-23	-30.9645
	AAPL	1999-11-24	-7.1066
	AAPL	1999-11-26	-5.4121
	AAPL	1999-11-29	-44.3969
	AAPL	1999-11-30	-37.4761
	AAPL	1999-12-01	-8.7591
	AAPL	1999-12-02	-1.906
	AAPL	1999-12-03	-2.0364
	AAPL	1999-12-06	-4.547
	AAPL	1999-12-07	-0.6441
	AAPL	1999-12-08	-30.1786
	AAPL	1999-12-09	-51.5152
	AAPL	1999-12-10	-60.6061
	AAPL	1999-12-13	-92.0989
	AAPL	1999-12-14	-99.4839
	AAPL	1999-12-15	-77.951
	AAPL	1999-12-16	-73.0883
	AAPL	1999-12-17	-66.8151
	AAPL	1999-12-20	-74.239

	wma_name	wma_day	wma_value
▶	AAPL	2000-01-14	0.7751
	AAPL	2000-01-21	0.7777
	AAPL	2000-01-28	0.7888
	AAPL	2000-02-04	0.787
	AAPL	2000-02-11	0.7936
	AAPL	2000-02-18	0.8001
	AAPL	2000-02-25	0.8084
	AAPL	2000-03-03	0.8157
	AAPL	2000-03-10	0.8434
	AAPL	2000-03-17	0.8629
	AAPL	2000-03-24	0.8818
	AAPL	2000-03-31	0.9173
	AAPL	2000-04-07	0.9448
	AAPL	2000-04-14	0.9633
	AAPL	2000-04-20	0.9468
	AAPL	2000-04-28	0.9378
	AAPL	2000-05-05	0.9416
	AAPL	2000-05-12	0.9257
	AAPL	2000-05-19	0.9039
	AAPL	2000-05-26	0.8648
	AAPL	2000-06-02	0.8212
	AAPL	2000-06-09	0.7904
	AAPL	2000-06-16	0.7699
	AAPL	2000-06-23	0.7469
	AAPL	2000-06-30	0.7495
	AAPL	2000-07-07	0.7517

	stock_name	stock_day	stock_low	stock_high	stock_open	stock_close	stock_volume	stock_dividend_amo...	stock_split_coeffici...
▶	AAPL	2022-07-12	145.05	148.45	145.76	145.86	77588800	0	1
	AAPL	2022-07-13	142.12	146.45	142.99	145.49	71185600	0	1
	AAPL	2022-07-14	143.25	148.95	144.08	148.47	78140700	0	1
	AAPL	2022-07-15	148.2	150.86	149.78	150.17	76259900	0	1
	AAPL	2022-07-18	146.7	151.57	150.74	147.07	81420900	0	1
	AAPL	2022-07-19	146.91	151.23	147.92	151	82982400	0	1
	AAPL	2022-07-20	150.37	153.72	151.12	153.04	64823400	0	1
	AAPL	2022-07-21	151.94	155.57	154.5	155.35	65086600	0	1
	AAPL	2022-07-22	153.41	156.28	155.39	154.09	66675400	0	1
	AAPL	2022-07-25	152.28	155.04	154.01	152.95	53623900	0	1
	AAPL	2022-07-26	150.8	153.085	152.265	151.6	55138700	0	1
	AAPL	2022-07-27	152.16	157.33	152.58	156.79	78620700	0	1
	AAPL	2022-07-28	154.41	157.64	156.98	157.35	81378700	0	1
	AAPL	2022-07-29	159.5	163.63	161.24	162.51	101787000	0	1
	AAPL	2022-08-01	160.89	163.59	161.01	161.51	67829400	0	1
	AAPL	2022-08-02	159.63	162.41	160.1	160.01	59907000	0	1
	AAPL	2022-08-03	160.75	166.59	160.84	166.13	82507500	0	1
	AAPL	2022-08-04	164.43	167.19	166.005	165.81	55474100	0	1
	AAPL	2022-08-05	163	165.85	163.21	165.35	56697000	0.23	1
	AAPL	2022-08-08	164.2	167.81	166.37	164.87	60362300	0	1
	AAPL	2022-08-09	163.25	165.82	164.02	164.92	63135500	0	1
	AAPL	2022-08-10	166.9	169.34	167.68	169.24	70170500	0	1
	AAPL	2022-08-11	168.19	170.99	170.06	168.49	57149200	0	1
	AAPL	2022-08-12	169.4	172.17	169.82	172.1	68039400	0	1
	AAPL	2022-08-15	171.345	173.39	171.52	173.19	54091700	0	1
	AAPL	2022-08-16	171.662	173.71	172.78	173.03	56377000	0	1

QUERY STATEMENTS WITH VIEWS:

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.

```
CREATE VIEW DI_ANALYSIS_VIEW AS
  SELECT
    *
 FROM
```

```
    minusdi t1
      INNER JOIN
    plusdi t2 ON MINUS_DI_name = PLUS_DI_name;



SELECT     MINUS_DI_name, MINUS_DI_value, PLUS_DI_value
FROM DI_ANALYSIS_VIEW
   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.

```
CREATE VIEW AROON_VIEW AS
   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;


SELECT   *
FROM AROON_VIEW
   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.

```
CREATE VIEW HT_VIEW AS
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;

SELECT * FROM HT_VIEW;
```

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.

```
CREATE VIEW MA_VIEW AS
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;

SELECT * FROM MA_VIEW;
```

- **Query:**
  What was the closing on Apple stock last year in November?

```
SELECT STOCK_DAY, STOCK_CLOSE
FROM MONTHLY
WHERE STOCK_DAY BETWEEN '2021-11-01' AND '2021-11-31';
```

- **Query:**

  What has been the lowest value on Apple stock?

SELECT stock_close, stock_day

from monthly

where stock_close = (SELECT MIN(stock_close) from monthly);

- **Query:**

  What was the exchange rate for Bitcoin(BTC) to USD from last year to this year?

  SELECT close_USD, currency_day

  from digital_currency_monthly

  where currency_day BETWEEN '2021-01-01' AND '2022-02-01';

- **Query:**

  What was the average stock value for Apple this month?

  SELECT AVG(stock_close)

  from daily_adjusted

  WHERE stock_day between '2022-12-01' AND '2022-12-31';

- **Query:**

  Show me the highest volume of AAPL Stock.

  SELECT stock_volume, stock_day

  from monthly

  where stock_volume = (SELECT MAX(stock_volume) from monthly);

- **Manually inserting data into a table:**

```
INSERT INTO daily_adjusted
VALUES ('AAPL', '2022-12-16', '141.77', '146.77', '145.77',
'143.77','77777800', '7', '7');

SELECT * FROM daily_adjusted;
```

JUPYTER NOTEBOOK WITH ALL PYTHON SCRIPTS AND VISUALISATION:

In [28]:
```
# IMPORTS
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')
```

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


```
[
  {
    "2022-12-14": {
      "1. open": "145.35",
      "2. high": "146.66",
      "3. low": "141.16",
      "4. close": "143.21",
      "5. adjusted close": "143.21",
      "6. volume": "82220264",
      "7. dividend amount": "0.0000",
      "8. split coefficient": "1.0"
    },
    "2022-12-13": {
      "1. open": "149.5",
      "2. high": "149.9692",
      "3. low": "144.24",
      "4. close": "145.47",
      "5. adjusted close": "145.47",
      "6. volume": "93886161",
```


In [3]:
```
# Checking the connection to MySQL
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')
```

```
Connection established to glance_at_finance
```

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

In [4]:
```
# DAILY ADJUSTED STOCK DATA

# 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))
```

```
Connection established to glance_at_finance
```

In [5]:
```
# 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))
```

```
Connection established to glance_at_finance
```

In [ ]:
```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')


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))
```

In [ ]:
```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))
```

In [ ]:
```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))
```

In [ ]:
```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))
```

In [12]:
```
# 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))
```

```
Connection established to glance_at_finance
{
  "symbol": "AAPL",
  "annualEarnings": [
    {
      "fiscalDateEnding": "2022-09-30",
      "reportedEPS": "6.11"
    },
    {
      "fiscalDateEnding": "2021-09-30",
      "reportedEPS": "5.62"
    },
    {
      "fiscalDateEnding": "2020-09-30",
      "reportedEPS": "3.27"
    },
    {
      "fiscalDateEnding": "2019-09-30",
      "reportedEPS": "2.98"
```

In [23]:
```
# 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 survivo

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)
```

```
['symbol', 'name', 'exchange', 'assetType', 'ipoDate', 'delistingDate', 'status']
['A', 'Agilent Technologies Inc', 'NYSE', 'Stock', '1999-11-18', 'null', 'Active']
['AA', 'Alcoa Corp', 'NYSE', 'Stock', '2016-10-18', 'null', 'Active']
['AAA', 'AXS FIRST PRIORITY CLO BOND ETF ', 'NYSE ARCA', 'ETF', '2020-09-09', 'null', 'Active']
['AAAU', 'Goldman Sachs Physical Gold ETF', 'BATS', 'ETF', '2018-08-15', 'null', 'Active']
['AAC', 'Ares Acquisition Corporation - Class A', 'NYSE', 'Stock', '2021-03-25', 'null', 'Active']
['AAC-U', 'Ares Acquisition Corporation - Units (1 Ord Share Class A & 1/5 War)', 'NYSE', 'Stock', '2021-02-02', 'null', 'Active']
['AAC-WS', 'Ares Acquisition Corporation - Warrants (01/01/9999)', 'NYSE', 'Stock', '2021-03-25', 'null', 'Active']
['AACG', 'ATA Creativity Global', 'NASDAQ', 'Stock', '2008-01-29', 'null', 'Active']
['AACI', 'Armada Acquisition Corp I', 'NASDAQ', 'Stock', '2021-11-10', 'null', 'Active']
['AACIU', 'Armada Acquisition Corp I - Units (1 Ord & 1/2 War)', 'NASDAQ', 'Stock', '2021-08-13', 'null', 'Active']
['AACIW', 'Armada Acquisition Corp I - Warrants (13/08/2026)', 'NASDAQ', 'Stock', '2021-11-11', 'null', 'Active']
['AADI', 'Aadi Bioscience Inc', 'NASDAQ', 'Stock', '2017-08-08', 'null', 'Active']
['AADR', 'AdvisorShares Dorsey Wright ADR ETF', 'NASDAQ', 'ETF', '2010-07-21', 'null', 'Active']
['AAIC', 'Arlington Asset Investment Corp - Class A', 'NYSE', 'Stock', '1997-12-23', 'null', 'Active']
['AAIC-P-B', 'Arlington Asset Investment Corp', 'NYSE', 'Stock', '2017-05-16', 'null', 'Active']
['AAIC-P-C', 'Arlington Asset Investment Corp', 'NYSE', 'Stock', '2019-03-06', 'null', 'Active']
['AAIN', 'Arlington Asset Investment Corp', 'NYSE', 'Stock', '2021-07-19', 'null', 'Active']
['AAL', 'American Airlines Group Inc', 'NASDAQ', 'Stock', '2005-09-27', 'null', 'Active']
```

In [24]:
```
# Earnings Calendar
# This API returns a list of company earnings expected in the next 3, 6, or 12 months.


import csv
import requests


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)
```

```
['AAN', 'Aarons Company Inc (The)', '2023-02-21', '2022-12-31', '0.01', 'USD']
['AAOI', 'Applied Optoelectronics Inc', '2023-02-22', '2022-12-31', '-0.31', 'USD']
['AAON', 'AAON Inc', '2023-02-27', '2022-12-31', '0.56', 'USD']
['AAP', 'Advance Auto Parts Inc', '2023-02-13', '2022-12-31', '2.43', 'USD']
['AAPL', 'Apple Inc', '2023-01-25', '2022-12-31', '1.99', 'USD']
['AAT', 'American Assets Trust Inc', '2023-02-06', '2022-12-31', '0.16', 'USD']
['AAVVF', 'AAVVF', '2023-02-22', '2022-12-31', '0.32', 'CAD']
['AAWH', 'AAWH', '2023-03-07', '2022-12-31', '-0.0317', 'USD']
['AAWW', 'Atlas Air Worldwide Holdings Inc', '2023-02-15', '2022-12-31', '5.41', 'USD']
['AB', 'AllianceBernstein Holding Lp', '2023-02-09', '2022-12-31', '0.56', 'USD']
['ABB', 'ABB Ltd', '2023-02-02', '2022-12-31', '0.3804', 'USD']
['ABBV', 'Abbvie Inc', '2023-01-31', '2022-12-31', '3.68', 'USD']
['ABC', 'Amerisource Bergen Corp', '2023-01-31', '2022-12-31', '2.61', 'USD']
['ABCB', 'Ameris Bancorp', '2023-01-25', '2022-12-31', '1.34', 'USD']
['ABCL', 'AbCellera Biologics Inc', '2023-02-22', '2022-12-31', '-0.01', 'USD']
['ABEV', 'Ambev S.A.', '2023-03-02', '2022-12-31', '0.05', 'BRL']
['ABG', 'Asbury Automotive Group Inc', '2023-02-13', '2022-12-31', '8.22', 'USD']
['ABILF', 'ABILF', '2023-03-01', '2022-12-31', '0', 'USD']
['ABIO', 'ARCA biopharma Inc', '2023-03-13', '2022-12-31', '', 'USD']
```

In [25]:
```
# IPO Calendar
# This API returns a list of IPOs expected in the next 3 months.

import csv
import requests


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)
```

```
['symbol', 'name', 'ipoDate', 'priceRangeLow', 'priceRangeHigh', 'currency', 'exchange']
['RAYA', 'Erayak Power Solution Group Inc. Class A Ordinary Shares', '2022-12-14', '4', '4', 'USD', 'NASDAQ']
['GEHCV', 'GE HealthCare Technologies Inc. Common Stock When-Issued', '2022-12-15', '0', '0', 'USD', 'NASDAQ']
['PRZO', 'ParaZero Technologies Ltd.', '2022-12-16', '4.25', '6.25', 'USD', 'NASDAQ']
['SODR', 'SONDORS Inc.', '2022-12-16', '8', '10', 'USD', 'NASDAQ']
['ATMVU', 'AlphaVest Acquisition Corp.', '2022-12-20', '10', '10', 'USD', 'NASDAQ']
['PWM', 'Prestige Wealth Inc.', '2022-12-30', '5.5', '6.5', 'USD', 'NASDAQ']
```

In [29]:
```
# 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()

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


```
{
  "Realtime Currency Exchange Rate": {
    "1. From_Currency Code": "USD",
    "2. From_Currency Name": "United States Dollar",
    "3. To_Currency Code": "JPY",
    "4. To_Currency Name": "Japanese Yen",
    "5. Exchange Rate": "135.44000000",
    "6. Last Refreshed": "2022-12-15 00:19:23",
    "7. Time Zone": "UTC",
    "8. Bid Price": "135.44000000",
    "9. Ask Price": "135.44000000"
  }
}
```

In [14]:
```
# 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()


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

```
Connection established to glance_at_finance
{
  "Meta Data": {
    "1. Information": "Forex Daily Prices (open, high, low, close)",
    "2. From Symbol": "EUR",
    "3. To Symbol": "USD",
    "4. Output Size": "Compact",
    "5. Last Refreshed": "2022-12-14 23:00:00",
    "6. Time Zone": "UTC"
  },
  "Time Series FX (Daily)": {
    "2022-12-14": {
      "1. open": "1.06270",
      "2. high": "1.06952",
      "3. low": "1.06170",
      "4. close": "1.06801"
    },
    "2022-12-13": {
      "1. open": "1.05355",
```

In [15]:
```
# 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))
```

```
Connection established to glance_at_finance
{
  "Meta Data": {
    "1. Information": "Forex Weekly Prices (open, high, low, close)",
    "2. From Symbol": "EUR",
    "3. To Symbol": "USD",
    "4. Last Refreshed": "2022-12-14 23:00:00",
    "5. Time Zone": "UTC"
  },
  "Time Series FX (Weekly)": {
    "2022-12-14": {
      "1. open": "1.05281",
      "2. high": "1.06952",
      "3. low": "1.05040",
      "4. close": "1.06800"
    },
    "2022-12-09": {
      "1. open": "1.05300",
      "2. high": "1.05948",
```

In [16]:
```
# 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))
```


```
{
  "Meta Data": {
    "1. Information": "Forex Monthly Prices (open, high, low, close)",
    "2. From Symbol": "EUR",
    "3. To Symbol": "USD",
    "4. Last Refreshed": "2022-12-14 23:00:00",
    "5. Time Zone": "UTC"
  },
  "Time Series FX (Monthly)": {
    "2022-12-14": {
      "1. open": "1.04041",
      "2. high": "1.06952",
      "3. low": "1.03906",
      "4. close": "1.06800"
    },
    "2022-11-30": {
      "1. open": "0.98818",
      "2. high": "1.04969",
      "3. low": "0.97280",
```


In [ ]:
```
# 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))
```

In [30]:
```
# 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))
```


```
{
  "name": "Real Gross Domestic Product",
  "interval": "annual",
  "unit": "billions of dollars",
  "data": [
    {
      "date": "2021-01-01",
      "value": "19609.812"
    },
    {
      "date": "2020-01-01",
      "value": "18509.143"
    },
    {
      "date": "2019-01-01",
      "value": "19036.052"
    },
    {
      "date": "2018-01-01",
```


In [31]:
```
# 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))
```


```
{
  "name": "Real Gross Domestic Product per Capita",
  "interval": "quarterly",
  "unit": "chained 2012 dollars",
  "data": [
    {
      "date": "2022-07-01",
      "value": "60135.0"
    },
    {
      "date": "2022-04-01",
      "value": "59756.0"
    },
    {
      "date": "2022-01-01",
      "value": "59877.0"
    },
    {
      "date": "2021-10-01",
```


In [32]:
```
# 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))
```


```
{
  "name": "10-Year Treasury Constant Maturity Rate",
  "interval": "monthly",
  "unit": "percent",
  "data": [
    {
      "date": "2022-11-01",
      "value": "3.89"
    },
    {
      "date": "2022-10-01",
      "value": "3.98"
    },
    {
      "date": "2022-09-01",
      "value": "3.52"
    },
    {
      "date": "2022-08-01",
```

In [33]:
```
# 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))
```


```
{
  "name": "Effective Federal Funds Rate",
  "interval": "monthly",
  "unit": "percent",
  "data": [
    {
      "date": "2022-11-01",
      "value": "3.78"
    },
    {
      "date": "2022-10-01",
      "value": "3.08"
    },
    {
      "date": "2022-09-01",
      "value": "2.56"
    },
    {
      "date": "2022-08-01",
```


In [34]:
```
# 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))
```


```
{
  "name": "Consumer Price Index for all Urban Consumers",
  "interval": "monthly",
  "unit": "index 1982-1984=100",
  "data": [
    {
      "date": "2022-11-01",
      "value": "297.711"
    },
    {
      "date": "2022-10-01",
      "value": "298.012"
    },
    {
      "date": "2022-09-01",
      "value": "296.808"
    },
    {
      "date": "2022-08-01",
```


In [35]:
```
# 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))
```


```
{
  "name": "Inflation - US Consumer Prices",
  "interval": "annual",
  "unit": "percent",
  "data": [
    {
      "date": "2021-01-01",
      "value": "4.69785886363739"
    },
    {
      "date": "2020-01-01",
      "value": "1.23358439630637"
    },
    {
      "date": "2019-01-01",
      "value": "1.81221007526015"
    },
    {
      "date": "2018-01-01",
```

In [36]:
```
# 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))
```


```
{
  "name": "Inflation Expectations",
  "interval": "monthly",
  "unit": "percent",
  "data": [
    {
      "date": "2022-10-01",
      "value": "5"
    },
    {
      "date": "2022-09-01",
      "value": "4.7"
    },
    {
      "date": "2022-08-01",
      "value": "4.8"
    },
    {
      "date": "2022-07-01",
```


In [38]:
```
# 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))
```


```
{
  "name": "Consumer Sentiment & Consumer Confidence",
  "interval": "monthly",
  "unit": "index 1966:Q1=100",
  "data": [
    {
      "date": "2022-10-01",
      "value": "59.9"
    },
    {
      "date": "2022-09-01",
      "value": "58.6"
    },
    {
      "date": "2022-08-01",
      "value": "58.2"
    },
    {
      "date": "2022-07-01",
```


In [39]:
```
# 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))
```


```
{
  "name": "Advance Retail Sales: Retail Trade",
  "interval": "monthly",
  "unit": "millions of dollars",
  "data": [
    {
      "date": "2022-10-01",
      "value": "597492"
    },
    {
      "date": "2022-09-01",
      "value": "576853"
    },
    {
      "date": "2022-08-01",
      "value": "613416"
    },
    {
      "date": "2022-07-01",
```

In [40]:
```
# 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))
```


```
{
  "name": "Manufacturer New Orders: Durable Goods",
  "interval": "monthly",
  "unit": "millions of dollars",
  "data": [
    {
      "date": "2022-10-01",
      "value": "273481"
    },
    {
      "date": "2022-09-01",
      "value": "289932"
    },
    {
      "date": "2022-08-01",
      "value": "279323"
    },
    {
      "date": "2022-07-01",
```


In [41]:
```
# 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 ar

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))
```


```
{
  "name": "Unemployment Rate",
  "interval": "monthly",
  "unit": "percent",
  "data": [
    {
      "date": "2022-11-01",
      "value": "3.7"
    },
    {
      "date": "2022-10-01",
      "value": "3.7"
    },
    {
      "date": "2022-09-01",
      "value": "3.5"
    },
    {
      "date": "2022-08-01",
```


In [42]:
```
# 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))
```


```
{
  "name": "Total Nonfarm Payroll",
  "interval": "monthly",
  "unit": "thousands of persons",
  "data": [
    {
      "date": "2022-11-01",
      "value": "154990"
    },
    {
      "date": "2022-10-01",
      "value": "154416"
    },
    {
      "date": "2022-09-01",
      "value": "153204"
    },
    {
      "date": "2022-08-01",
```

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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

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))
```

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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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))
```

In [ ]:
```
# 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 [ ]:
```
# VISUALIZATIONS
```

In [2]:
```
# Plotting the AAPL stock data
import matplotlib.pyplot as plt
import pandas as pd

# Reading the data
df = pd.read_csv('/Users/varadmurtymohod/Downloads/GAF Database/AAPL.csv')
type(df)

# Viewing the data
pd.set_option("display.max.columns", None)
df.head(10)
```

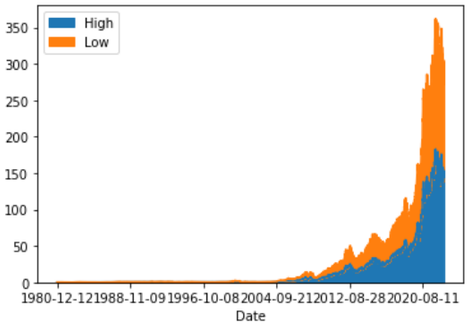
Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.099874	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094663	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087715	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.089886	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092492	73449600
5	1980-12-19	0.126116	0.126674	0.126116	0.126116	0.098137	48630400
6	1980-12-22	0.132254	0.132813	0.132254	0.132254	0.102913	37363200
7	1980-12-23	0.137835	0.138393	0.137835	0.137835	0.107256	46950400
8	1980-12-24	0.145089	0.145647	0.145089	0.145089	0.112901	48003200
9	1980-12-26	0.158482	0.159040	0.158482	0.158482	0.123323	55574400

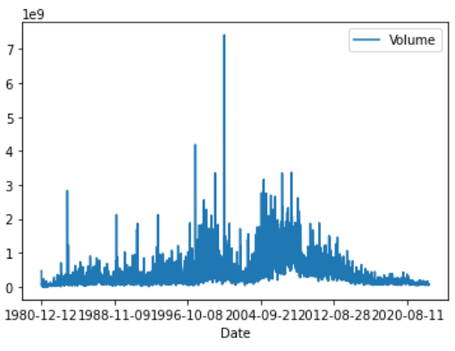
In [30]:
```
%matplotlib
```

```
Using matplotlib backend: MacOSX
```

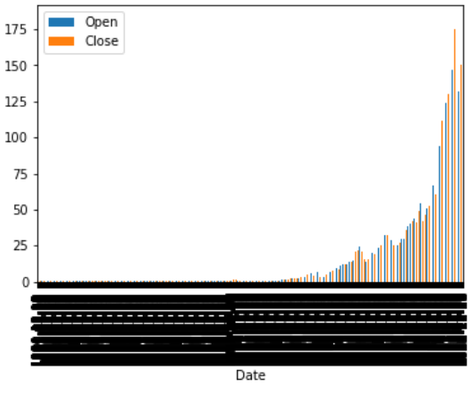
In [15]:
```
df.plot.area(x="Date", y=["High", "Low"])
plt.show()
# Displays the plot (might open in an exterior Python window)
```



In [14]:
```
df.plot.line(x="Date", y=["Volume"])
plt.show()
# Displays the plot (might open in an exterior Python window)
```



In [22]:
```
df.plot.bar(x="Date", y=["Open", "Close"])
plt.show()
# Displays the plot (might open in an exterior Python window)
```



In [ ]: