**## DOWNLOAD THE HISTORICAL STOCK DATA**

import yfinance as yf

import pandas as pd

from datetime import datetime

# CREATE TICKER INSTANCE FOR A STOCK

stock\_symbol = input("Enter the stock symbol: ")

stock\_ticker = yf.Ticker(stock\_symbol)

# Define the arguments for the history function

start\_date = '2020-01-01'

end\_date = '2023-05-10'

# GET TODAYS DATE AND CONVERT IT TO A STRING WITH YYYY-MM-DD FORMAT (YFINANCE EXPECTS THAT FORMAT)

stock\_symbol\_hist = stock\_ticker.history(start=start\_date, end=end\_date, interval=interval)

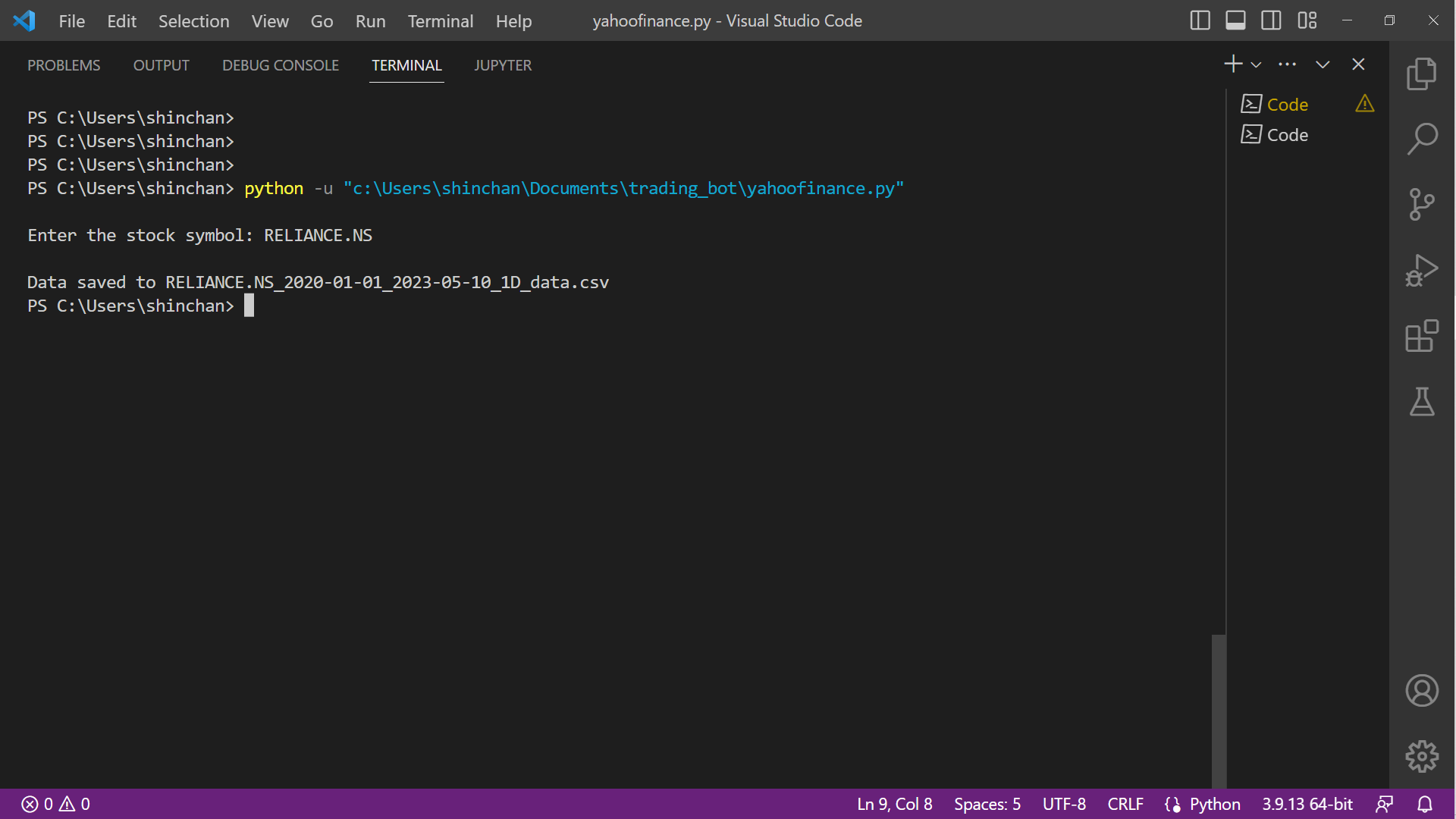
# Save the data to a CSV file

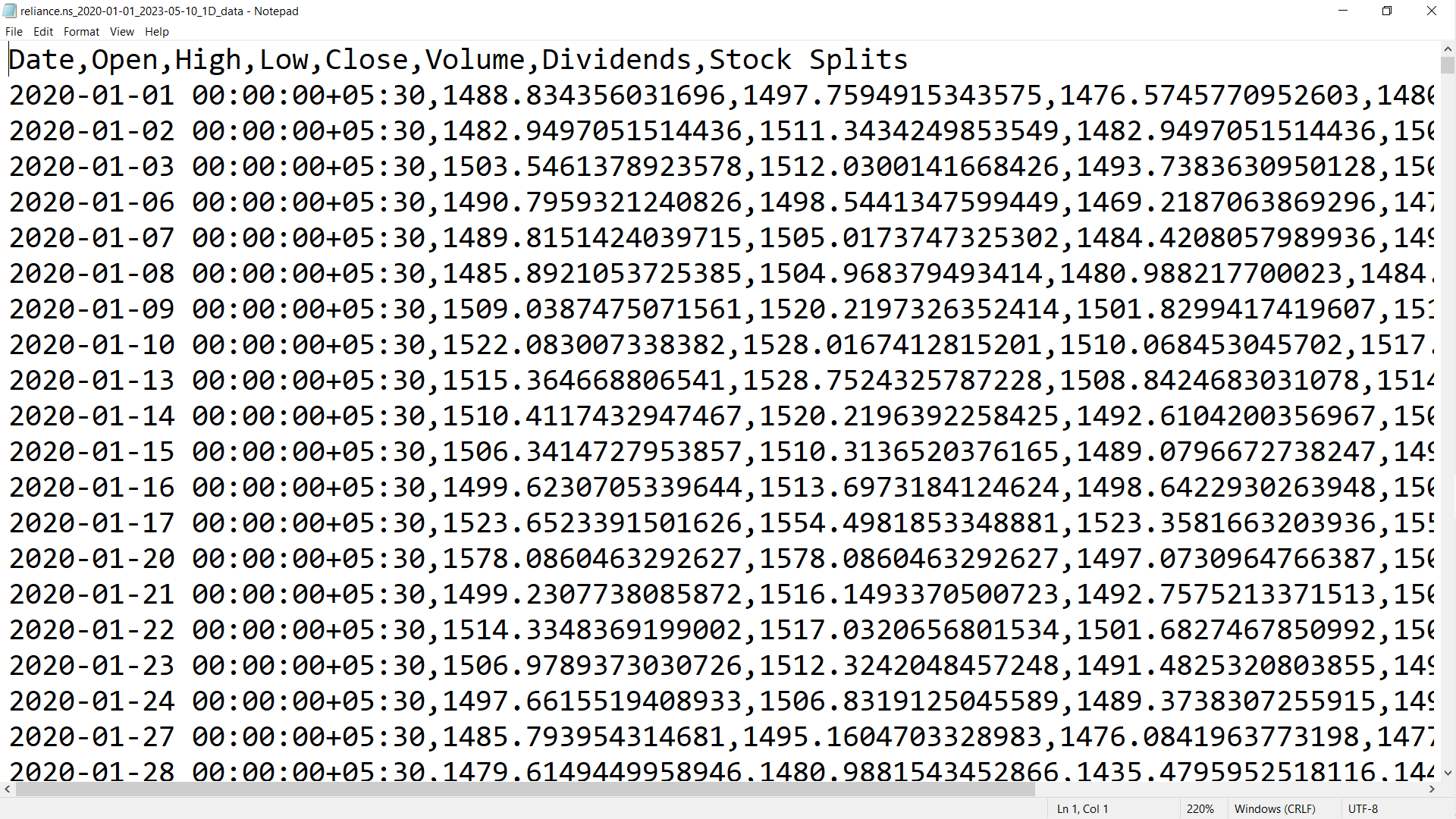
filename = stock\_symbol + '\_' + start\_date + '\_' + end\_date + '\_' + interval + '\_data.csv'

stock\_symbol\_hist.to\_csv(filename)

print('Data saved to', filename)

**OUTPUT:**



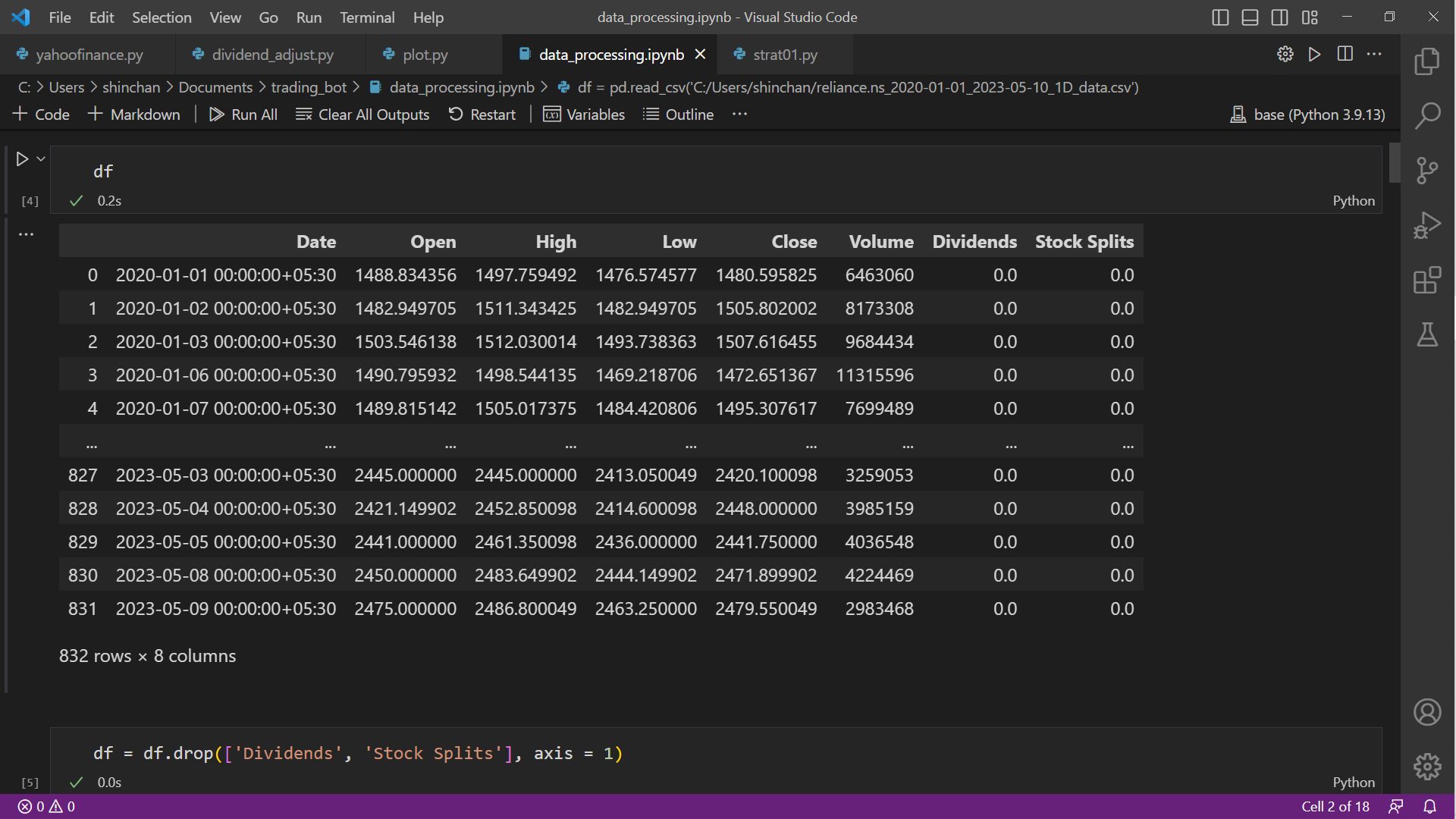


**## CLEAN AND PROCESS THE DATA**

import pandas as pd

df = pd.read\_csv('C:/Users/shinchan/reliance.ns\_2020-01-01\_2023-05-10\_1D\_data.csv')

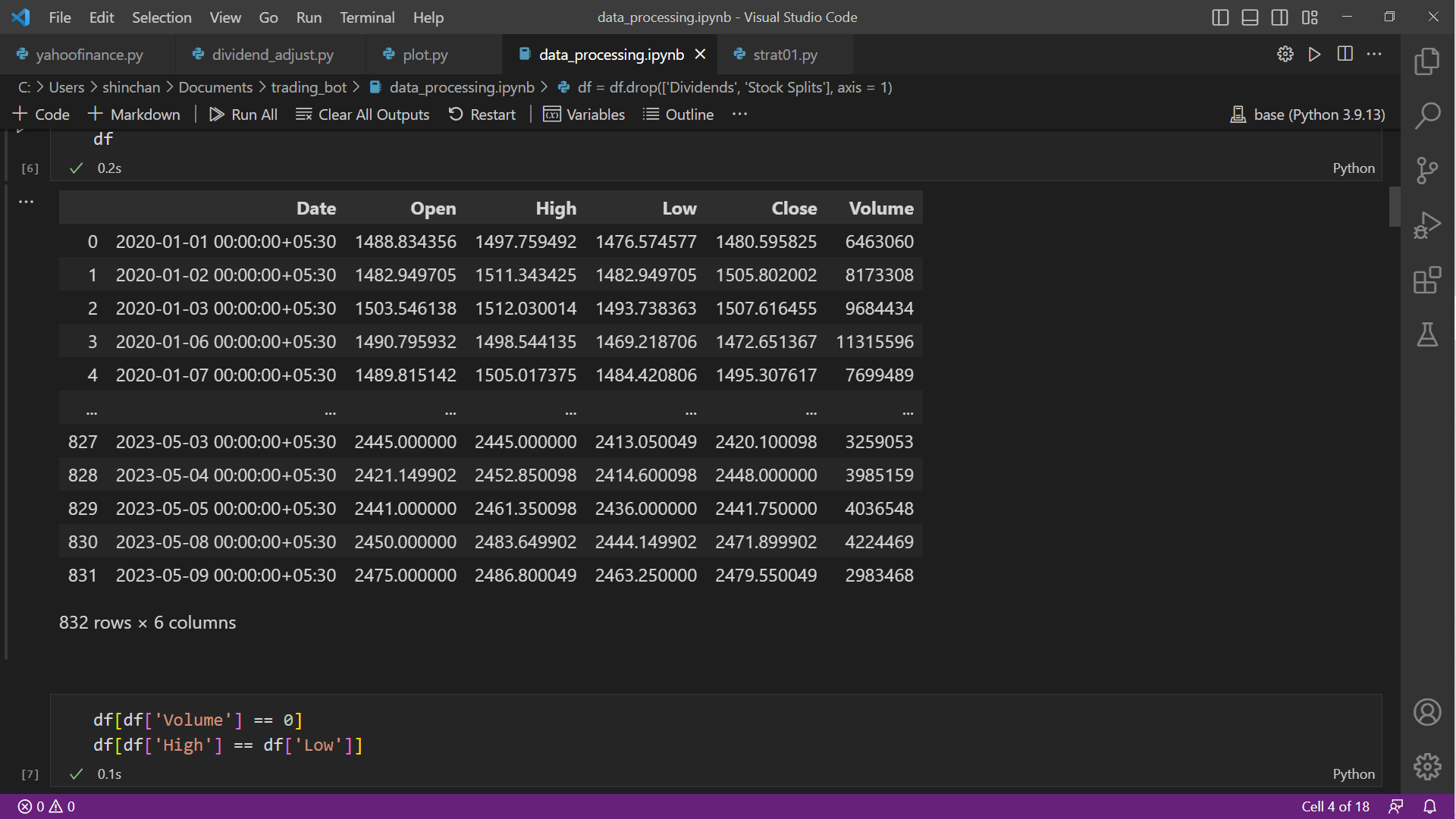
df



# FILTER THE COLUMNS NOT BEING USED

df = df.drop(['Dividends', 'Stock Splits'], axis = 1)

df



# FILTER THE NULL DATA

df\_filtered = df[df['High'] != df['Low']].copy()

# PLOT THE GRAPH

data = df[0:100]

import plotly.graph\_objects as go

fig = go.Figure([go.Candlestick(

x = data.index,

open = data['Open'],

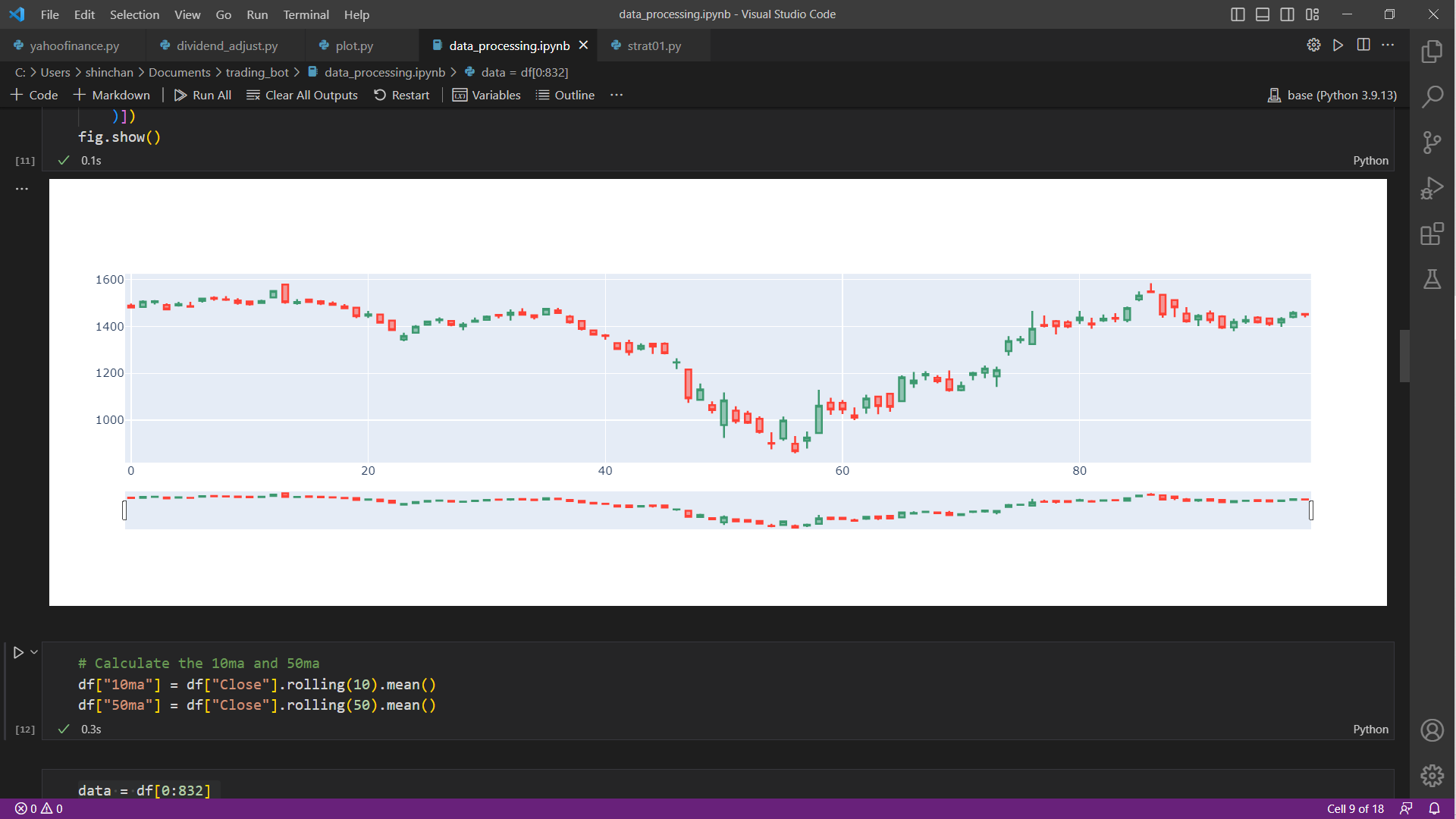
close = data['Close'],

high = data['High'],

low = data['Low']

)])

fig.show()



**## PERFORM TECHNICAL ANALYSIS ON DATA**

# CALCULATE MOVING AVERAGE INDICATOR (10ma and 50ma)

df["10ma"] = df["Close"].rolling(10).mean()

df["50ma"] = df["Close"].rolling(50).mean()

# PLOT THE INDIACATORS ON THE STOCK GRAPH

data = df[0:500]

fig = go.Figure()

fig.add\_trace(go.Candlestick(

x = data.index,

open = data['Open'],

close = data['Close'],

high = data['High'],

low = data['Low']

))

fig.add\_trace(go.Scatter(

x=data.index,

y=data['50ma'],

name='50ma'

))

fig.add\_trace(go.Scatter(

x=data.index,

y=data['10ma'],

name='10ma'

))

fig.show()



**## BUILD A EXPERT SYSTEM WHICH CAN BACKTEST TRADING BASED ON MA INDICATOR**

**AND GIVE US PERCENTAGE GAINS/LOSSES**

data = df

# Initialize buy and sell columns

data['buy'] = 0

data['sell'] = 0

# Initialize successful\_trade and failed\_trade lists

successful\_trade = []

failed\_trade = []

total\_trade = []

# Loop through the data

for i in range(1, len(data)):

# Check for buy signal

if data['10ma'][i] > data['50ma'][i] and data['10ma'][i-1] <= data['50ma'][i-1]:

data['buy'][i] = 1

# Check for successful or failed trade

if data['Close'][i] \* 1.05 <= max(data['High'][i:]):

successful\_trade.append(i)

elif data['Close'][i] \* 0.98 >= min(data['Low'][i:]):

failed\_trade.append(i)

# Check for sell signal

if data['10ma'][i] < data['50ma'][i] and data['10ma'][i-1] >= data['50ma'][i-1]:

data['sell'][i] = -1

# USING THE RISK TO REWARD RATIO OF 2:5 (%) TO EXECUTE TRADES

if data['Close'][i] \* 0.95 >= min(data['Low'][i:]):

successful\_trade.append(i)

elif data['Close'][i] \* 1.02 <= max(data['High'][i:]):

failed\_trade.append(i)

## Print results

print('Successful trades:', len(successful\_trade))

print('Failed trades:', len(failed\_trade))

print('Total percentage profit:', (len(successful\_trade)\*5) - (len(failed\_trade)\*2), '%')

**OUTPUT:**

