master\_file\_path = "C:\\Users\\User\\Dropbox\\SQL Stock Market Data\\Python Code from ChatGPT\\Master File\\CompanyMaster.csv"

directory = "C:\\Users\\User\\Dropbox\\SQL Stock Market Data\\Data Files\\" # Replace with the path to your directory

column\_order = ['NSE\_Code','BSE\_Code','Date','NSE\_Delivery','BSE\_Delivery']

import os

import csv

import pandas as pd

def search\_nse\_code(directory, nse\_code, no\_days):

# Create an empty DataFrame to store the results

nse\_df = pd.DataFrame(columns=['NSE\_Code', 'Date','NSE\_Delivery'])

dely\_segment = ['EQ','BE','SM','ST']

file\_names = sorted([file for file in os.listdir(directory) if file.endswith('.csv') and file.startswith('NSE Delivery')], reverse=True)[:no\_days]

for file\_name in file\_names:

file\_path = os.path.join(directory, file\_name)

date = file\_name[13:19]

with open(file\_path, 'r') as file:

reader = csv.reader(file)

found\_match = False # Flag variable to track the match

for i, line in enumerate(reader, start=1):

if i >= 5: # Start reading from line 5

line\_str = ','.join(line)

data = line\_str.split(',')

if data[2] == nse\_code and data[3] in dely\_segment:

dely = data[5]

nse\_df = nse\_df.append({'NSE\_Code': nse\_code, 'Date': date, 'NSE\_Delivery':dely}, ignore\_index=True)

found\_match = True

break

if found\_match:

continue

return nse\_df

def search\_bse\_code(directory, bse\_code, no\_days):

# Create an empty DataFrame to store the results

bse\_df = pd.DataFrame(columns=['BSE\_Code', 'Date','BSE\_Delivery'])

dely\_segment = ['EQ','BE','SM','ST']

file\_names = sorted([file for file in os.listdir(directory) if file.endswith('.txt') and file.startswith('BSE Gross Deliverable ')], reverse=True)[:no\_days]

for file\_name in file\_names:

file\_path = os.path.join(directory, file\_name)

date = file\_name[22:28]

with open(file\_path, 'r') as file:

reader = csv.reader(file)

found\_match = False # Flag variable to track the match

for i, line in enumerate(reader, start=1):

if i >= 2: # Start reading from line 5

line\_str = ','.join(line)

data = line\_str.split('|')

if data[1] == bse\_code:

dely = data[2]

bse\_df = bse\_df.append({'BSE\_Code': bse\_code, 'Date': date, 'BSE\_Delivery':dely}, ignore\_index=True)

found\_match = True

break

if found\_match:

continue

return bse\_df

def find\_corresponding\_codes(master\_file\_path, user\_input):

nse\_code = ""

bse\_code = ""

user\_input = user\_input.lower() # Convert user input to lowercase for comparison

with open(master\_file\_path, 'r') as file:

reader = csv.reader(file)

for row in reader:

if row[1].lower() == user\_input: # NSE code match (case-insensitive)

nse\_code = row[1]

bse\_code = str(row[2])

break

elif row[2].lower() == user\_input: # BSE code match (case-insensitive)

nse\_code = row[1]

bse\_code = str(row[2])

break

return nse\_code, bse\_code

def format\_value(value):

return '{:,.0f}'.format(value)

def format\_decimal(x):

return '{:,.2f}'.format(x)

def search\_nse\_price(directory, nse\_code, no\_days):

# Create an empty DataFrame to store the results

nse\_df = pd.DataFrame(columns=['NSE\_Code', 'Date','Price'])

dely\_segment = ['EQ','BE','SM','ST']

file\_names = sorted([file for file in os.listdir(directory) if file.lower().endswith(('CSV','.csv')) and file.startswith('NSE EQ')], reverse=True)[:no\_days]

for file\_name in file\_names:

file\_path = os.path.join(directory, file\_name)

date = file\_name[7:13]

with open(file\_path, 'r') as file:

reader = csv.reader(file)

found\_match = False # Flag variable to track the match

for row in reader:

data\_list = []

data\_list.append(row)

if data\_list[0][0] == nse\_code and data\_list[0][1] in dely\_segment:

price = data\_list[0][5]

nse\_df = nse\_df.append({'NSE\_Code': nse\_code, 'Date': date, 'Price':price}, ignore\_index=True)

found\_match = True

break

if found\_match:

continue

return nse\_df

def search\_bse\_price(directory, bse\_code, no\_days):

# Create an empty DataFrame to store the results

bse\_df = pd.DataFrame(columns=['BSE\_Code', 'Date','Price'])

dely\_segment = ['EQ','BE','SM','ST']

file\_names = sorted([file for file in os.listdir(directory) if file.lower().endswith(('CSV','csv')) and file.startswith('EQ ISIN')], reverse=True)[:no\_days]

for file\_name in file\_names:

file\_path = os.path.join(directory, file\_name)

date = file\_name[8:14]

with open(file\_path, 'r') as file:

reader = csv.reader(file)

found\_match = False # Flag variable to track the match

for row in reader:

data\_list = []

data\_list.append(row)

if data\_list[0][0] == bse\_code :

price = data\_list[0][7]

bse\_df = bse\_df.append({'BSE\_Code': bse\_code, 'Date': date, 'Price':price}, ignore\_index=True)

found\_match = True

break

if found\_match:

continue

return bse\_df

###\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### displaying information for multiple scrips

scrips = ['RITES','KPIL','GAIL','SJVN']

no\_days = 15

formatted\_dfs = []

for i in scrips:

nse\_code, bse\_code = find\_corresponding\_codes(master\_file\_path, i)

nse\_result = search\_nse\_code(directory, nse\_code,no\_days)

bse\_result = search\_bse\_code(directory, bse\_code,no\_days)

if nse\_result.empty:

merged\_df = pd.merge(bse\_result, nse\_result, on='Date', how='left')

merged\_df['NSE\_Delivery'] = merged\_df['NSE\_Delivery'].fillna(0)

merged\_df['NSE\_Code'] = merged\_df['NSE\_Code'].fillna("")

# Check if bse\_result is empty

elif bse\_result.empty:

merged\_df = pd.merge(nse\_result, bse\_result, on='Date', how='left')

merged\_df['BSE\_Delivery'] = merged\_df['BSE\_Delivery'].fillna(0)

merged\_df['BSE\_Code'] = merged\_df['BSE\_Code'].fillna("")

# Neither nse\_result nor bse\_result is empty

else:

merged\_df = pd.merge(nse\_result, bse\_result, on='Date', how='inner')

merged\_df = merged\_df[column\_order]

merged\_df['NSE\_Delivery'] = merged\_df['NSE\_Delivery'].astype(int)

merged\_df['BSE\_Delivery'] = merged\_df['BSE\_Delivery'].astype(int)

merged\_df['Total\_Delivery'] = merged\_df['NSE\_Delivery'] + merged\_df['BSE\_Delivery']

nse\_price = search\_nse\_price(directory,nse\_code,no\_days)

nse\_price['Price'] = nse\_price['Price'].astype(float)

if nse\_price.empty:

bse\_price = search\_bse\_price(directory,bse\_code,no\_days)

bse\_price['Price'] = bse\_price['Price'].astype(float)

final\_df = pd.merge(merged\_df, bse\_price, on=['Date','BSE\_Code'], how='inner')

else:

final\_df = pd.merge(merged\_df, nse\_price, on=['Date','NSE\_Code'], how = 'inner')

##format the df for final printing/display

formatted\_df = final\_df.style.format({'NSE\_Delivery': format\_value, 'BSE\_Delivery': format\_value,

'Total\_Delivery': format\_value, 'Price' : format\_decimal})

#formatted\_df

# Append the formatted DataFrame to the list

#formatted\_dfs.append(formatted\_df)

# Print the formatted DataFrame

#print(f"Formatted DataFrame for {i}:")

#print(formatted\_df)

#print("\n") # Add a new line for separation

print(f"Formatted DataFrame for {i}:")

display(formatted\_df)

print("\n") # Add a new line for separation

###\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##getting user input

default\_value = 15

user\_input = input("Enter NSE code or BSE code: ")

no\_days = input(f"No of days for which data required:(default:{default\_value})")

while not no\_days.isdigit() or int(no\_days) <= 0:

print("Invalid input. Please enter a positive numeric value.")

no\_days = input("Enter a positive numeric value for no of days: ")

#"Enter a number (default: {default\_value}): "

no\_days = int(no\_days)

#### finding corresponding codes of nse and bse

nse\_code, bse\_code = find\_corresponding\_codes(master\_file\_path, user\_input)

if nse\_code and bse\_code:

print("Corresponding codes found:")

print(f"NSE code: {nse\_code}")

print(f"BSE code: {bse\_code}")

else:

print("Code not found in the master file.")

print(f"NSE code: {nse\_code}")

print(f"BSE code: {bse\_code}")

### finding nse and bse delivery information

nse\_result = search\_nse\_code(directory, nse\_code,no\_days)

bse\_result = search\_bse\_code(directory, bse\_code,no\_days)

# Perform inner join on 'Date' field based on delivery info found

if nse\_result.empty:

merged\_df = pd.merge(bse\_result, nse\_result, on='Date', how='left')

merged\_df['NSE\_Delivery'] = merged\_df['NSE\_Delivery'].fillna(0)

merged\_df['NSE\_Code'] = merged\_df['NSE\_Code'].fillna("")

# Check if bse\_result is empty

elif bse\_result.empty:

merged\_df = pd.merge(nse\_result, bse\_result, on='Date', how='left')

merged\_df['BSE\_Delivery'] = merged\_df['BSE\_Delivery'].fillna(0)

merged\_df['BSE\_Code'] = merged\_df['BSE\_Code'].fillna("")

# Neither nse\_result nor bse\_result is empty

else:

merged\_df = pd.merge(nse\_result, bse\_result, on='Date', how='inner')

#### changing column order of merged df (column order defined in cell 1) and making delivery value as int

#### adding total delivery column

merged\_df = merged\_df[column\_order]

merged\_df['NSE\_Delivery'] = merged\_df['NSE\_Delivery'].astype(int)

merged\_df['BSE\_Delivery'] = merged\_df['BSE\_Delivery'].astype(int)

merged\_df['Total\_Delivery'] = merged\_df['NSE\_Delivery'] + merged\_df['BSE\_Delivery']

#### search for price info in nse/bse files and merge with delivery data

nse\_price = search\_nse\_price(directory,nse\_code,no\_days)

nse\_price['Price'] = nse\_price['Price'].astype(float)

if nse\_price.empty:

bse\_price = search\_bse\_price(directory,bse\_code,no\_days)

bse\_price['Price'] = bse\_price['Price'].astype(float)

final\_df = pd.merge(merged\_df, bse\_price, on=['Date','BSE\_Code'], how='inner')

else:

final\_df = pd.merge(merged\_df, nse\_price, on=['Date','NSE\_Code'], how = 'inner')

##format the df for final printing/display

formatted\_df = final\_df.style.format({'NSE\_Delivery': format\_value, 'BSE\_Delivery': format\_value,

'Total\_Delivery': format\_value, 'Price' : format\_decimal})

formatted\_df