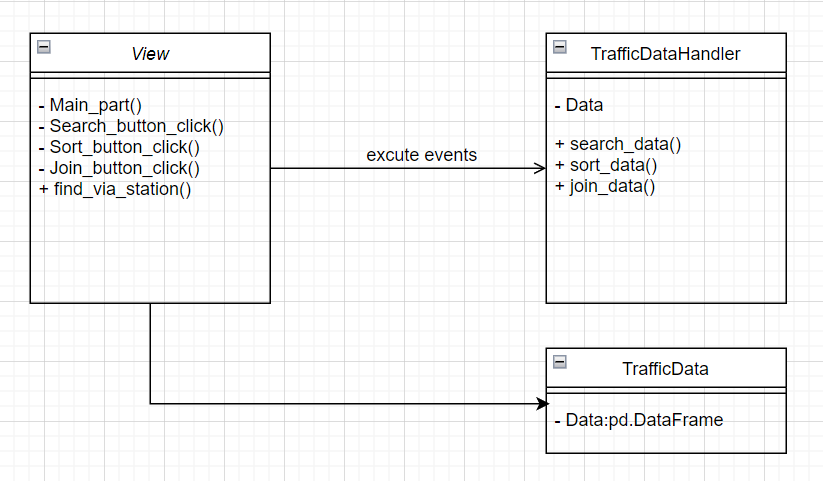
**MSDM5051 Project2**

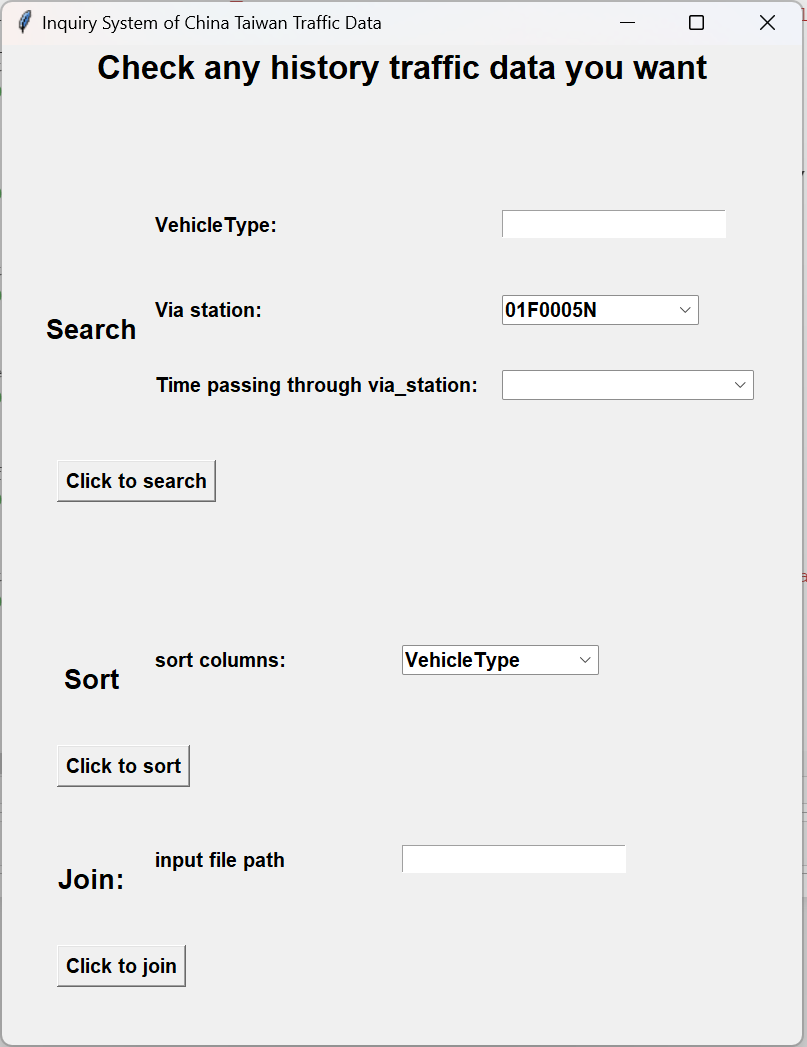
**CHEN Longyin,FENG Zekai,ZHANG Mingtao**

1. **UML Design**

Our UML design is shown as follows:



1. **Interface design**

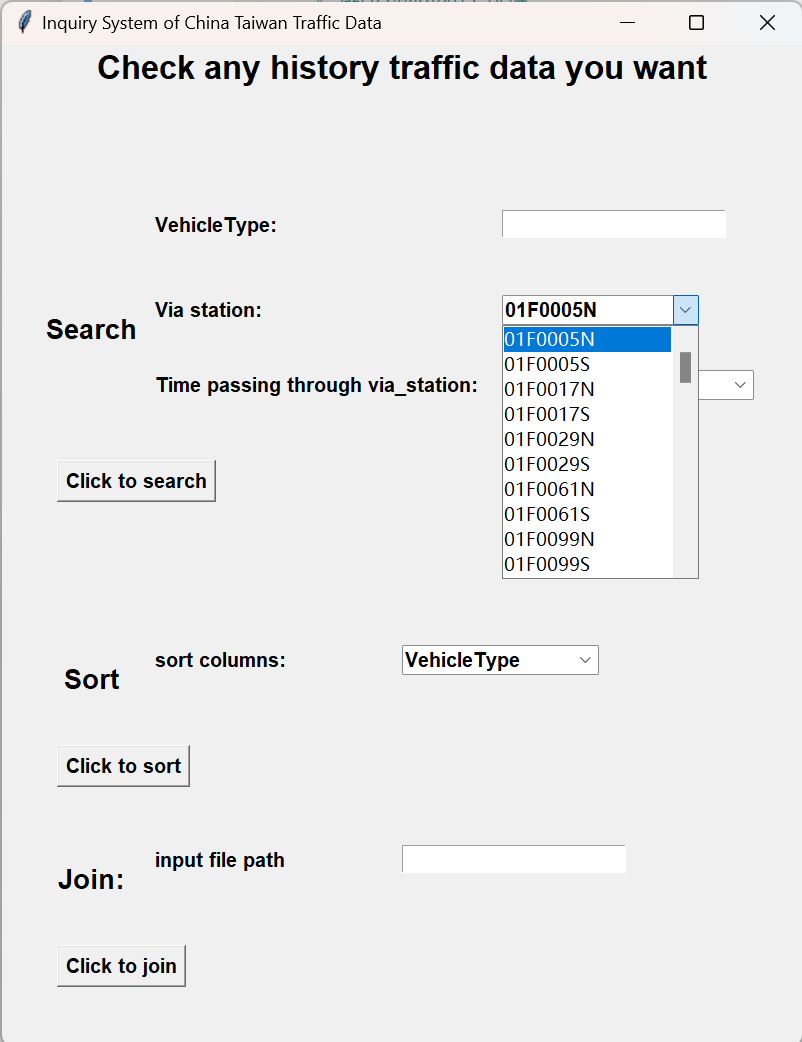


1. **Demonstration of how to use the program**

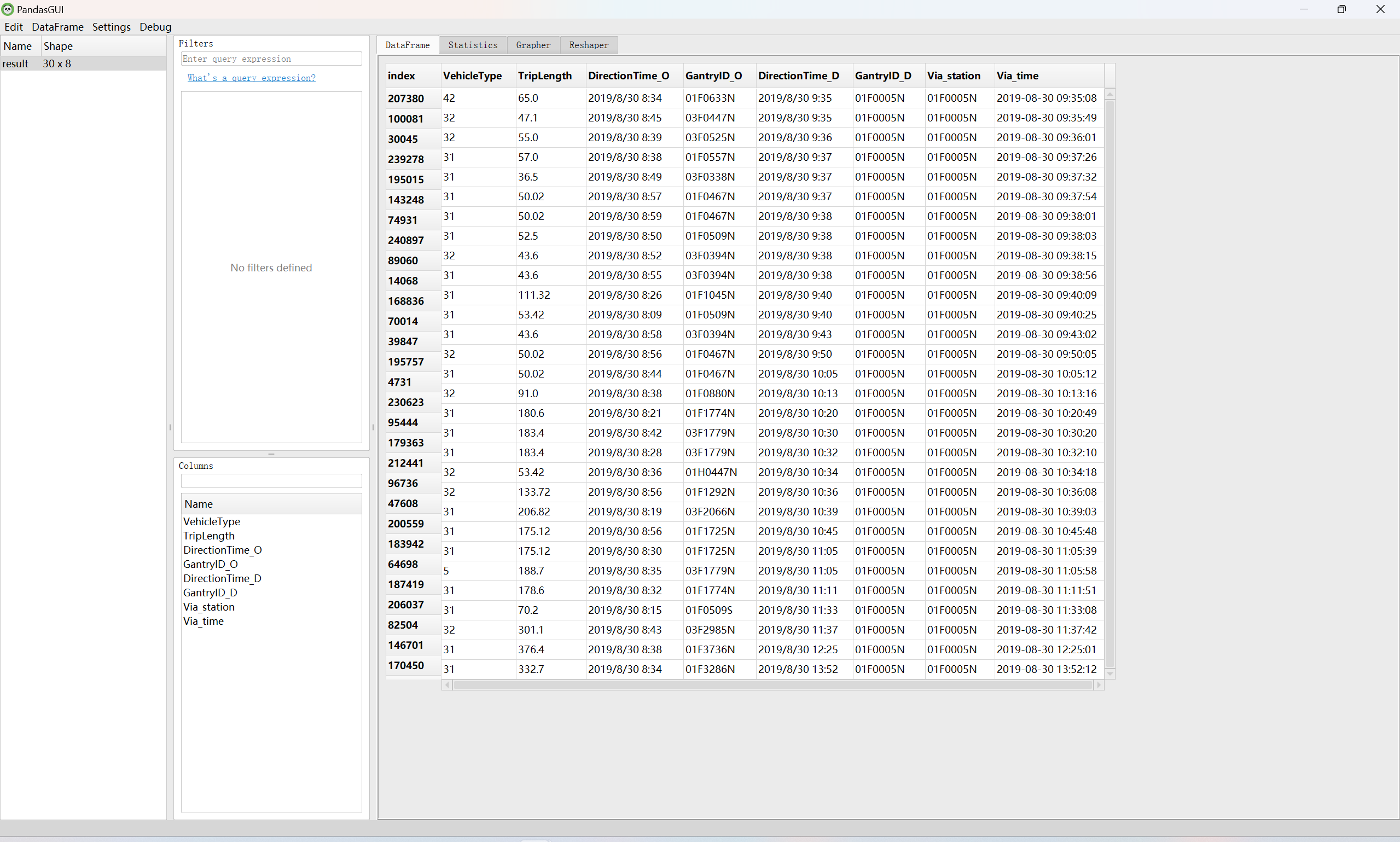
Our interactive interface consists of three components: Search, Sort, and Join. We aim to provide a comprehensive functionality that assists users in querying the historical data of a particular vehicle model passing through a specific station after a given time. This process is similar to checking real-time traffic data during our daily commutes. For instance, if I'm taking a bus, I would like to know how long it will take for a specific bus to arrive at my current location after the current time, helping me determine which bus to take.

* 1. **Search**

The search functionality includes three options: vehicle type (defaulting to all vehicle types if not specified), via stations (defaulting to the first station in the list), and time of transit through the station (defaulting to all data after 8:00 AM).



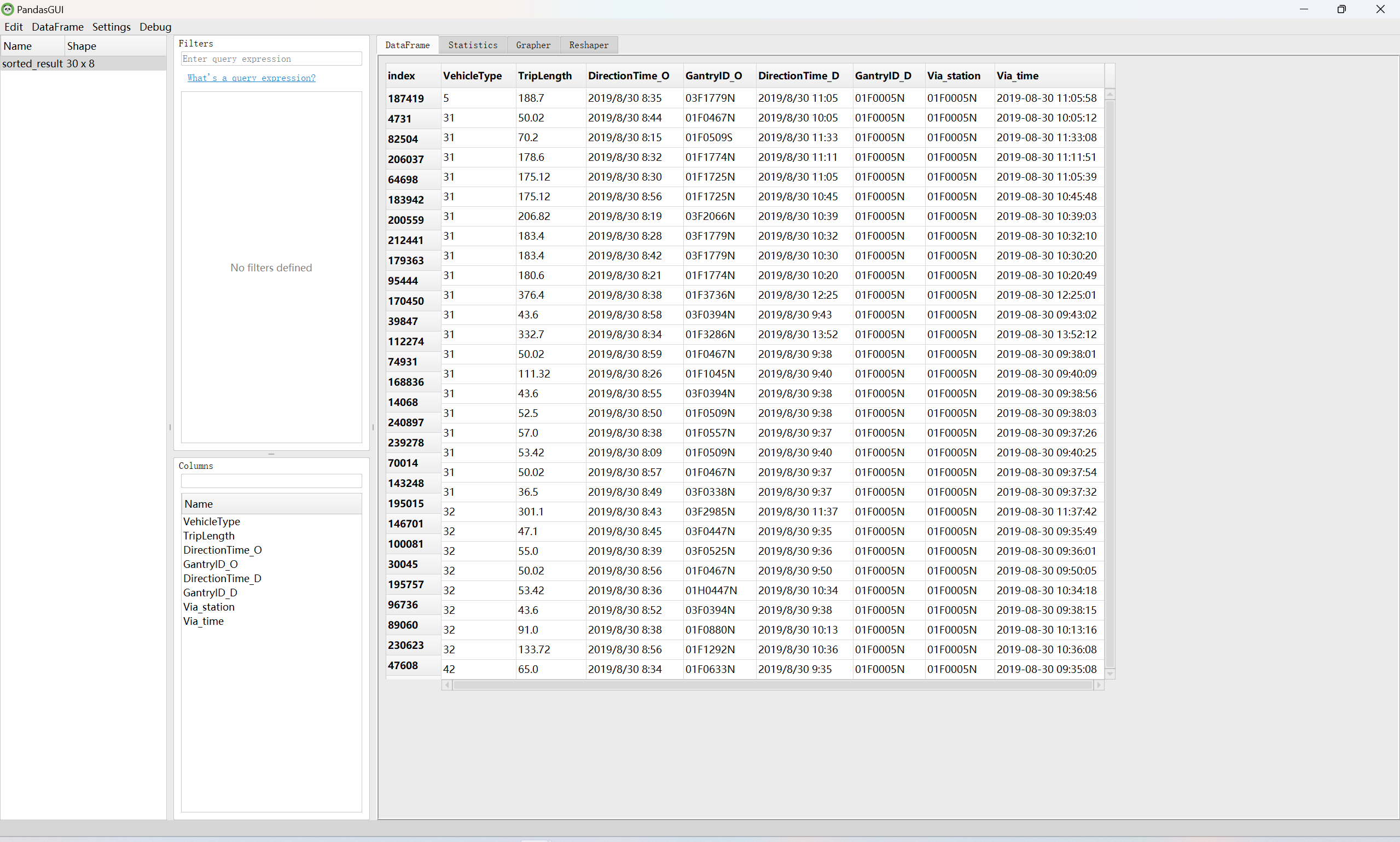
Here's an example using the default inputs for the search:



* 1. **Sort**

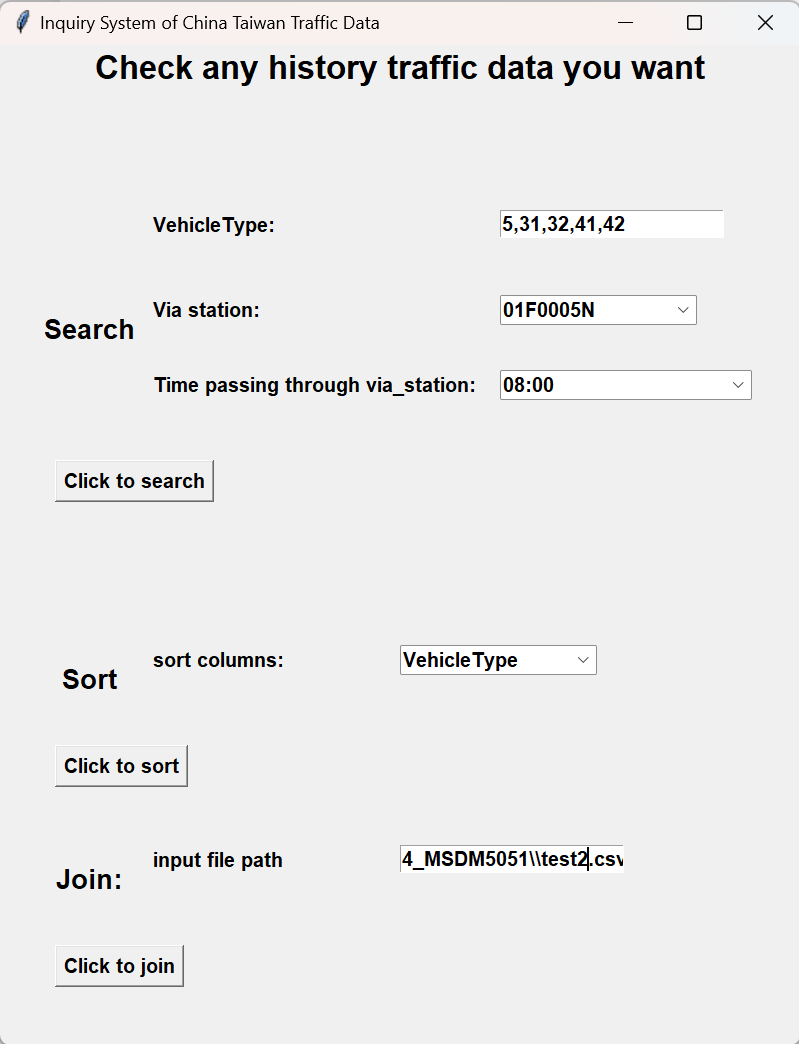
The sorting functionality includes one option: the element to be sorted. You can choose to sort by vehicle type, transit time through the station, or the overall length of the journey.

Here's an example using the default inputs for sorting:

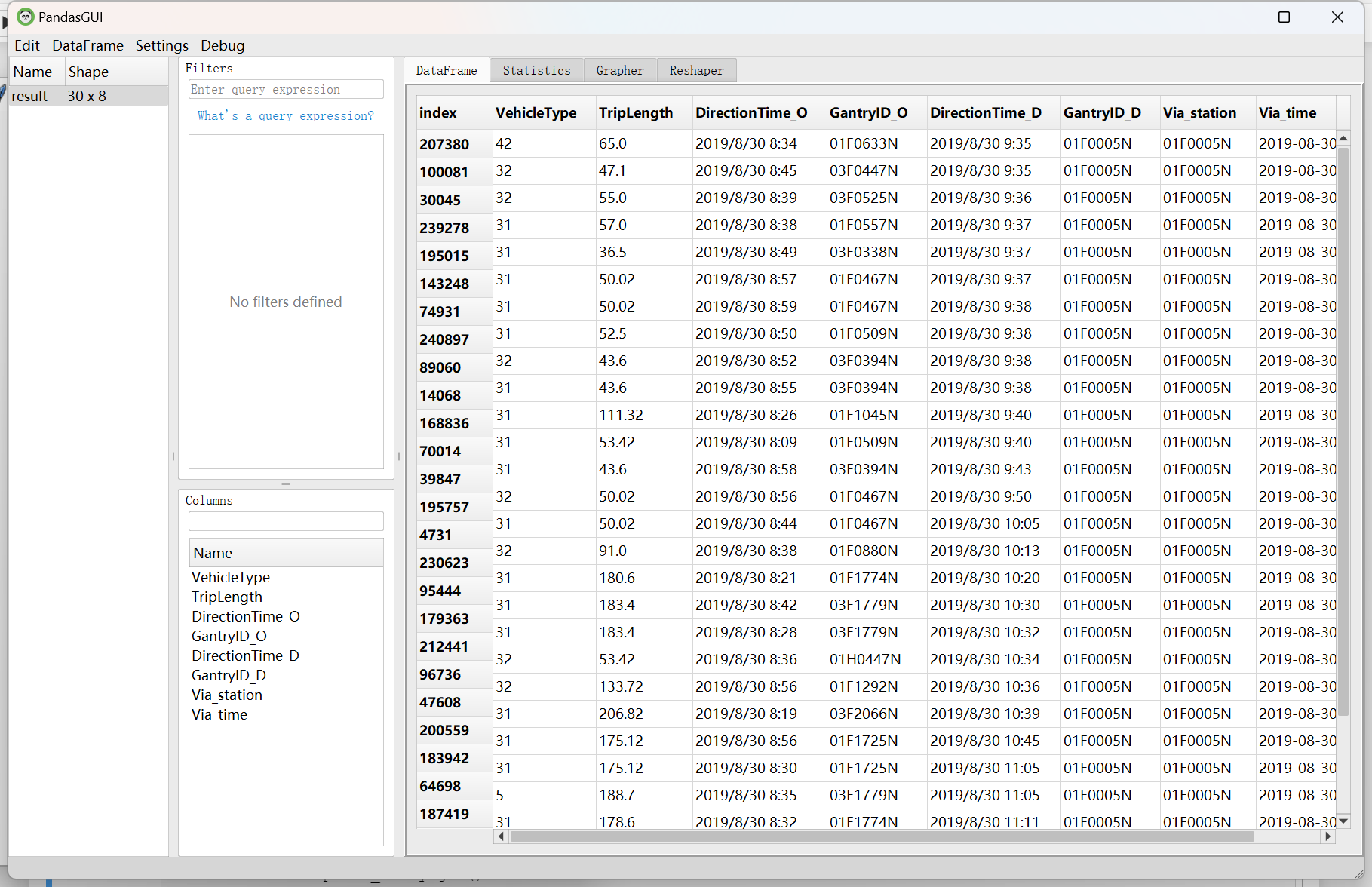


* 1. **Join**

The join functionality includes one option: the input of the file path you want to merge, which can be either a relative or absolute path. After entering the path and "Click to join," it will merge with the default traffic dataset and display the results of the transit stations you wish to query.



Here's an example showcasing the query results after merging subset 1 and subset 2 to form the complete traffic dataset:



1. **Raw Code**

# -\*- coding: gbk -\*-

import csv

import time

import timeit

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import copy

from datetime import datetime

import tkinter as tk

from tkinter import ttk

from tabulate import tabulate

from pandasgui import show

## pip install ttkwidgets

from ttkwidgets.autocomplete import AutocompleteCombobox

# traffic\_data = pd.read\_csv('D:\Python\_code\hw\_4\_MSDM5051\TDCS\_M06A\_20190830\_080000.csv',

# header = None,

# names = ['VehicleType','DirectionTime\_O','GantryID\_O','DirectionTime\_D','GantryID\_D','TripLength','TripEnd','TripInformation'])

traffic\_data = pd.read\_csv('D:\\Python\_code\\hw\_4\_MSDM5051\\test1.csv',

header = None,

names = ['VehicleType','DirectionTime\_O','GantryID\_O','DirectionTime\_D','GantryID\_D','TripLength','TripEnd','TripInformation'])

def find\_via\_station(data,station\_node):

data['index1'] = data[data['TripInformation'].str.contains(station\_node)]['TripInformation'].apply(lambda x:x.find(station\_node))-20

data['index2'] = data[data['TripInformation'].str.contains(station\_node)]['TripInformation'].apply(lambda x:x.find(station\_node))-1

data = data.fillna(0)

data['index1'] = data['index1'].astype(int)

data['index2'] = data['index2'].astype(int)

data['Via\_time'] = data.apply(lambda x:x['TripInformation'][x['index1']:x['index2']],axis = 1)

return data[data['Via\_time'].apply(len) > 0]

def link\_start\_ultra():

# 处理函数，输出一个数据集

def main\_part():

# 若空则初始化

if timepoint\_entry.get() == "":

timepoint\_entry.insert(tk.END, "08:00")

if bus\_entry.get() == "":

bus\_entry.insert(tk.END, "5,31,32,41,42")

if path\_entry.get() != "":

join\_data = pd.read\_csv(path\_entry.get(),

header = None,

names = ['VehicleType','DirectionTime\_O','GantryID\_O','DirectionTime\_D','GantryID\_D','TripLength','TripEnd','TripInformation'])

traffic\_data\_join = pd.concat([traffic\_data,join\_data])

print(join\_data.head())

else:

traffic\_data\_join = traffic\_data

# 获取用户输入的起点站和终点站

buslist = [int(num) for num in bus\_entry.get().split(",")]

via\_station = via\_station\_entry.get()

timepoint = timepoint\_entry.get()

filtered1=traffic\_data\_join[traffic\_data\_join.loc[:,"VehicleType"].isin(buslist)]

filtered2 = find\_via\_station(filtered1,via\_station)

filtered2["Via\_station"] = via\_station

sorted\_object = filtered2.sort\_values(by=["Via\_time","VehicleType"],ascending=True)

time\_obj = datetime.strptime(timepoint, "%H:%M")

formatted\_time = time\_obj.strftime("%H:%M:%S")

filtered3 = sorted\_object[sorted\_object.loc[:,"Via\_time"] > formatted\_time]

if path\_entry.get() != "":

result = filtered3.loc[:,["VehicleType","TripLength","DirectionTime\_O","GantryID\_O","DirectionTime\_D","GantryID\_D","Via\_station","Via\_time"]].tail(30)

else:

result = filtered3.loc[:,["VehicleType","TripLength","DirectionTime\_O","GantryID\_O","DirectionTime\_D","GantryID\_D","Via\_station","Via\_time"]].tail(30)

return result

# 输出查询结果

def search\_button\_click():

result = main\_part()

show(result)

# 输出排序结果

def sort\_button\_click():

result = main\_part()

indi = combo\_sort\_column.get()

sorted\_result = result.sort\_values(by=indi, ascending=True)

show(sorted\_result)

# 输出排序结果

def join\_button\_click():

result = main\_part()

show(result)

# 创建主窗口

root = tk.Tk()

root.title("Inquiry System of China Taiwan Traffic Data ")

# 设置窗口大小

root.geometry('800x1000')

# 标题

title = tk.Label(root, text='Check any history traffic data you want', font=('Arial', 25, 'bold'))#, width=20, height=3

title.pack()

# 子标题1

subtitle1 = tk.Label(root, text='Search', font=('Arial', 20, 'bold'), width=10, height=2)

subtitle1.place(x=2, y=250)

# 创建公交标签和输入框

bus\_label = tk.Label(root, text="VehicleType:", font=('Arial', 15, 'bold'))

bus\_label.place(x=150, y=165)

# bus\_label.pack()

bus\_entry = tk.Entry(root, font=('Arial', 15, 'bold'))

bus\_entry.place(x=500, y=165)

# bus\_entry.pack()

# 创建途径站标签和输入框

via\_station\_label = tk.Label(root, text="Via station:", font=('Arial', 15, 'bold'))

via\_station\_label.place(x = 150, y=250)

# start\_station\_label.pack()

all\_station\_list = list(np.sort(list(set(traffic\_data.loc[:,"GantryID\_O"].unique().tolist() + traffic\_data.loc[:,"GantryID\_D"].unique().tolist()))))

via\_station\_entry = AutocompleteCombobox(completevalues= all\_station\_list, font=('Arial', 15, 'bold'), width=15)

default\_station\_column = all\_station\_list[0]

via\_station\_entry.set(default\_station\_column)

via\_station\_entry.place(x=500, y=250)

# via\_station\_entry.pack()

# 创建时间点标签和输入框

timepoint\_label = tk.Label(root, text="Time passing through via\_station:", font=('Arial', 15, 'bold'))

timepoint\_label.place(x = 150, y=325)

# timepoint\_label.pack()

start\_time = "08:00"

end\_time = "09:00"

interval = 1 # 间隔时间，单位为分钟

start\_hour, start\_minute = map(int, start\_time.split(":"))

end\_hour, end\_minute = map(int, end\_time.split(":"))

start\_total\_minutes = start\_hour \* 60 + start\_minute

end\_total\_minutes = end\_hour \* 60 + end\_minute

time\_list = []

current\_minutes = start\_total\_minutes

while current\_minutes <= end\_total\_minutes:

hour = current\_minutes // 60

minute = current\_minutes % 60

time\_str = f"{hour:02d}:{minute:02d}"

time\_list.append(time\_str)

current\_minutes += interval

timepoint\_entry = AutocompleteCombobox(completevalues=time\_list, font=('Arial', 15, 'bold'))

timepoint\_entry.place(x=500, y=325)

# timepoint\_entry.pack()

# 创建搜索按钮

search\_button = tk.Button(root, text="Click to search", command=search\_button\_click, font=('Arial', 15, 'bold'))

search\_button.place(x=55, y=415)

# search\_button.pack()

# 排序部分的布局

subtitle2 = tk.Label(root, text='Sort', font=('Arial', 20, 'bold'), width=10, height=2)

subtitle2.place(x=2, y=600)

# 排序列标签和下拉框

label\_sort\_column = tk.Label(root,text='sort columns:', font=('Arial', 15, 'bold'))

label\_sort\_column.place(x=150, y=600)

COLUMN\_NAMES = ["VehicleType","TripLength","Via\_time"]

combo\_sort\_column = AutocompleteCombobox(completevalues=COLUMN\_NAMES, font=('Arial', 15, 'bold'), width=15)

default\_sort\_column = COLUMN\_NAMES[0]

combo\_sort\_column.set(default\_sort\_column)

combo\_sort\_column.place(x=400, y=600)

# 排序按钮

sort\_button = tk.Button(root, text="Click to sort", command=sort\_button\_click, font=('Arial', 15, 'bold'))

sort\_button.place(x=55, y=700)

# 创建公交标签和输入框

path\_label = tk.Label(root, text="Join:", font=('Arial', 20, 'bold'), width=10, height=2)

path\_label.place(x=2, y = 800)

# 排序部分的布局

subtitle3 = tk.Label(root, text='input file path', font=('Arial', 15, 'bold'))

subtitle3.place(x=150, y = 800)

# 创建合并标签与输入框

path\_entry = tk.Entry(root, font=('Arial', 15, 'bold'))

path\_entry.place(x=400, y= 800)

# 创建合并按钮

merge\_botton = tk.Button(root, text="Click to join", command = join\_button\_click, font=('Arial', 15, 'bold'))

merge\_botton.place(x=55, y=900)

# 启动主循环

root.mainloop()

link\_start\_ultra()