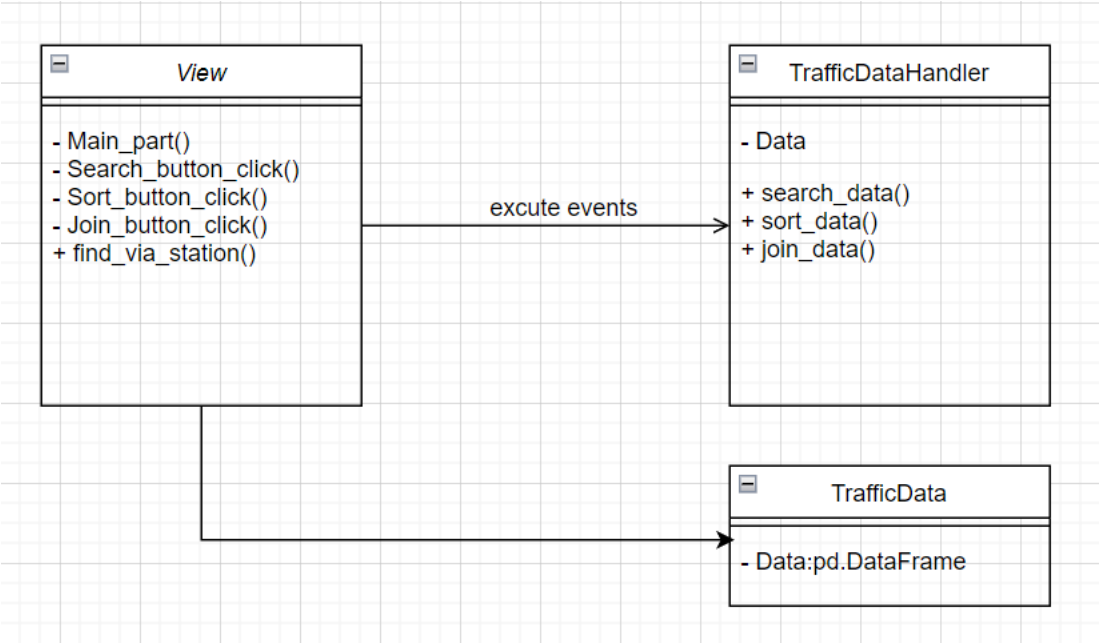
MSDM5051 Project2

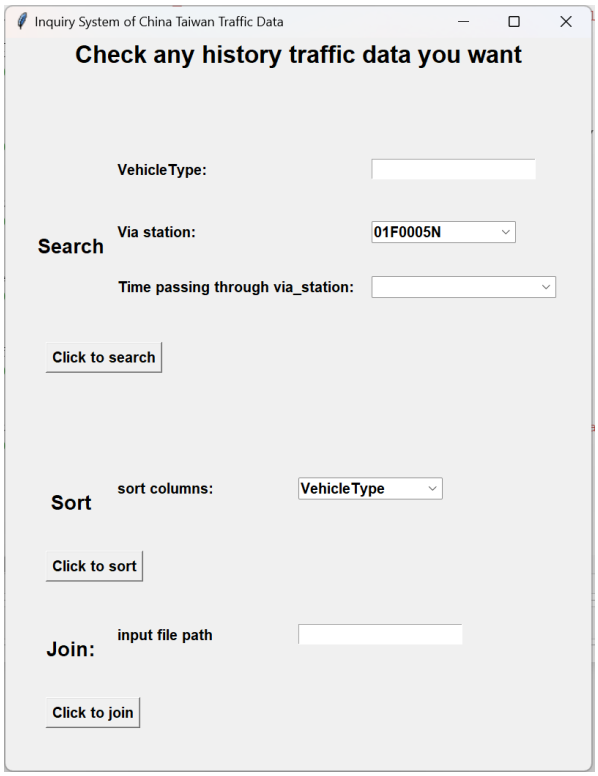
CHEN Longyin,FENG Zekai,ZHANG Mingtao

1. UML Design

Our UML design is shown as follows:



2. Interface design

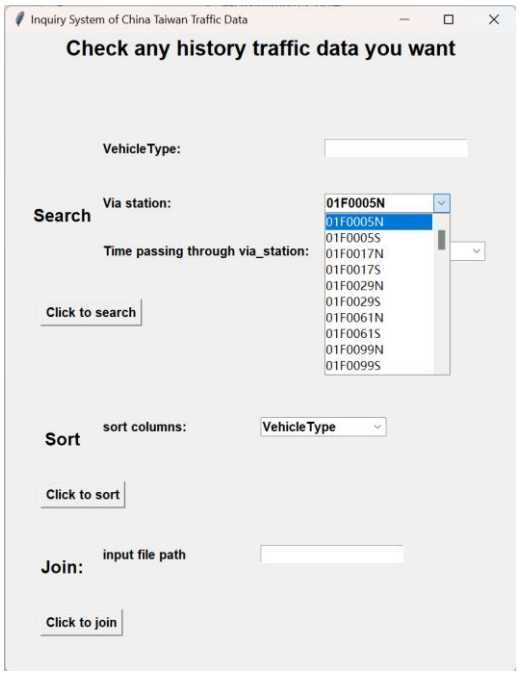


3. Demonstration of how to use the program

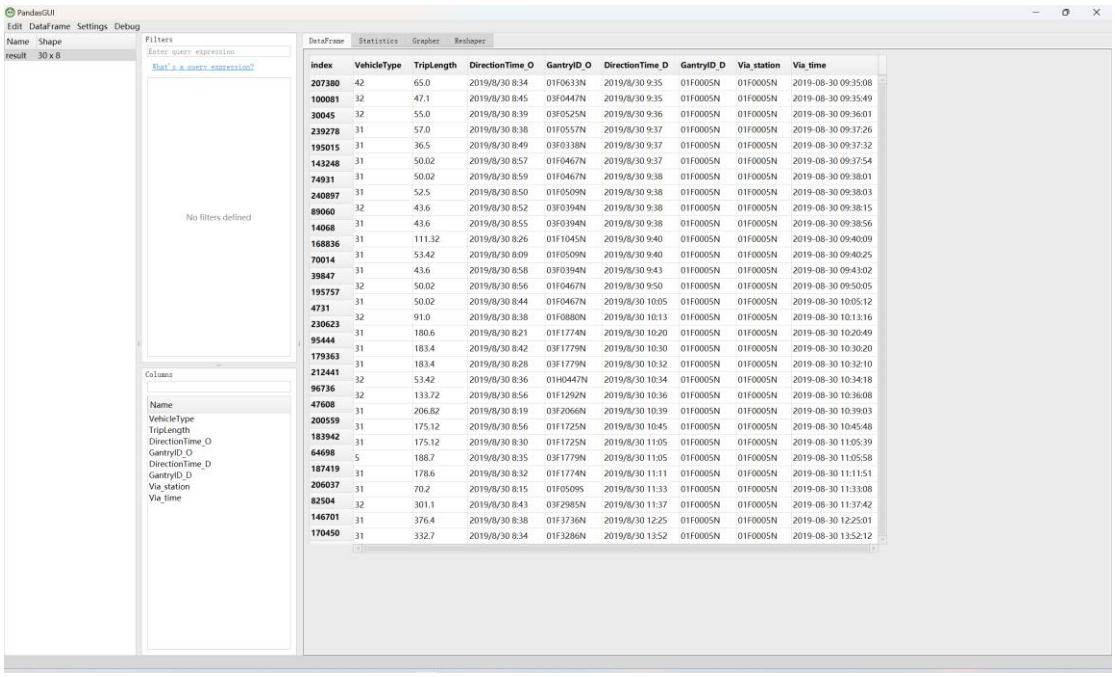
Our interactive interface consists of three components: Search, Sort, and Join. We aim to provide a comprehensive functionality that assists users in querying the historical data of a particular vehicle model passing through a specific station after a given time. This process is similar to checking real-time 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.

3.1. 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:

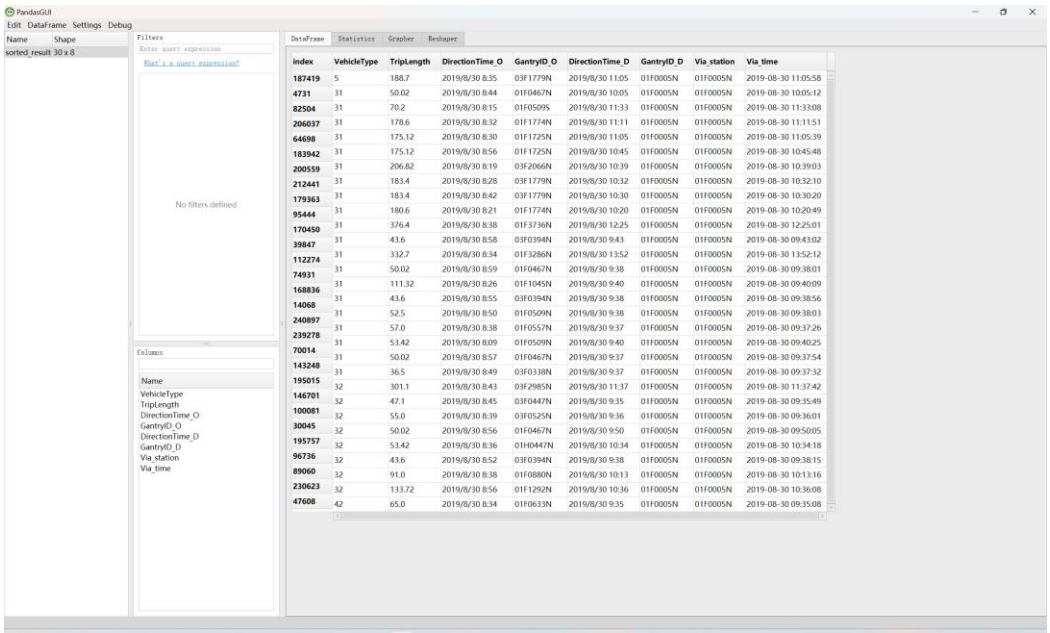


index	VehicleType	TripLength	DirectionTime_O	GantryID_O	DirectionTime_D	GantryID_D	Via_station	Via_time
207380	42	65.0	2019/8/30 8:34	01F0633N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30 09:35:08
100081	32	47.1	2019/8/30 8:45	03F0447N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30 09:35:49
30045	32	55.0	2019/8/30 8:39	03F0525N	2019/8/30 9:36	01F0005N	01F0005N	2019-08-30 09:36:01
239278	31	57.0	2019/8/30 8:38	01F0557N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:26
195015	31	36.5	2019/8/30 8:49	03F0338N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:32
143248	31	50.02	2019/8/30 8:57	01F0467N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:54
74931	31	50.02	2019/8/30 8:59	01F0467N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:01
240897	31	52.5	2019/8/30 8:50	01F0509N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:03
89060	32	43.6	2019/8/30 8:52	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:15
14068	31	43.6	2019/8/30 8:55	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:56
168836	31	111.32	2019/8/30 8:26	01F1045N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30 09:40:09
70014	31	53.42	2019/8/30 8:09	01F0509N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30 09:40:25
39847	31	43.6	2019/8/30 8:58	03F0394N	2019/8/30 9:43	01F0005N	01F0005N	2019-08-30 09:43:02
195757	32	50.02	2019/8/30 8:56	01F0467N	2019/8/30 9:50	01F0005N	01F0005N	2019-08-30 09:50:05
4731	31	50.02	2019/8/30 8:44	01F0467N	2019/8/30 10:05	01F0005N	01F0005N	2019-08-30 10:05:12
230623	32	91.0	2019/8/30 8:38	01F0880N	2019/8/30 10:13	01F0005N	01F0005N	2019-08-30 10:13:16
95444	31	180.6	2019/8/30 8:21	01F1774N	2019/8/30 10:20	01F0005N	01F0005N	2019-08-30 10:20:49
179363	31	183.4	2019/8/30 8:42	03F1779N	2019/8/30 10:30	01F0005N	01F0005N	2019-08-30 10:30:20
212441	31	183.4	2019/8/30 8:28	03F1779N	2019/8/30 10:32	01F0005N	01F0005N	2019-08-30 10:32:10
96736	32	53.42	2019/8/30 8:36	01H0447N	2019/8/30 10:34	01F0005N	01F0005N	2019-08-30 10:34:18
47608	32	133.72	2019/8/30 8:56	01F1292N	2019/8/30 10:36	01F0005N	01F0005N	2019-08-30 10:36:08
200559	31	206.82	2019/8/30 8:19	03F2066N	2019/8/30 10:39	01F0005N	01F0005N	2019-08-30 10:39:03
183942	31	175.12	2019/8/30 8:56	01F1725N	2019/8/30 10:45	01F0005N	01F0005N	2019-08-30 10:45:48
64698	31	175.12	2019/8/30 8:30	01F1725N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30 11:05:39
187419	5	188.7	2019/8/30 8:35	03F1779N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30 11:05:58
206037	31	178.6	2019/8/30 8:32	01F1774N	2019/8/30 11:11	01F0005N	01F0005N	2019-08-30 11:11:51
82504	31	70.2	2019/8/30 8:15	01F0509S	2019/8/30 11:33	01F0005N	01F0005N	2019-08-30 11:33:08
146701	32	301.1	2019/8/30 8:43	03F2985N	2019/8/30 11:37	01F0005N	01F0005N	2019-08-30 11:37:42
170450	31	376.4	2019/8/30 8:38	01F3736N	2019/8/30 12:25	01F0005N	01F0005N	2019-08-30 12:25:01
	31	332.7	2019/8/30 8:34	01F3286N	2019/8/30 13:52	01F0005N	01F0005N	2019-08-30 13:52:12

3.2. 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.

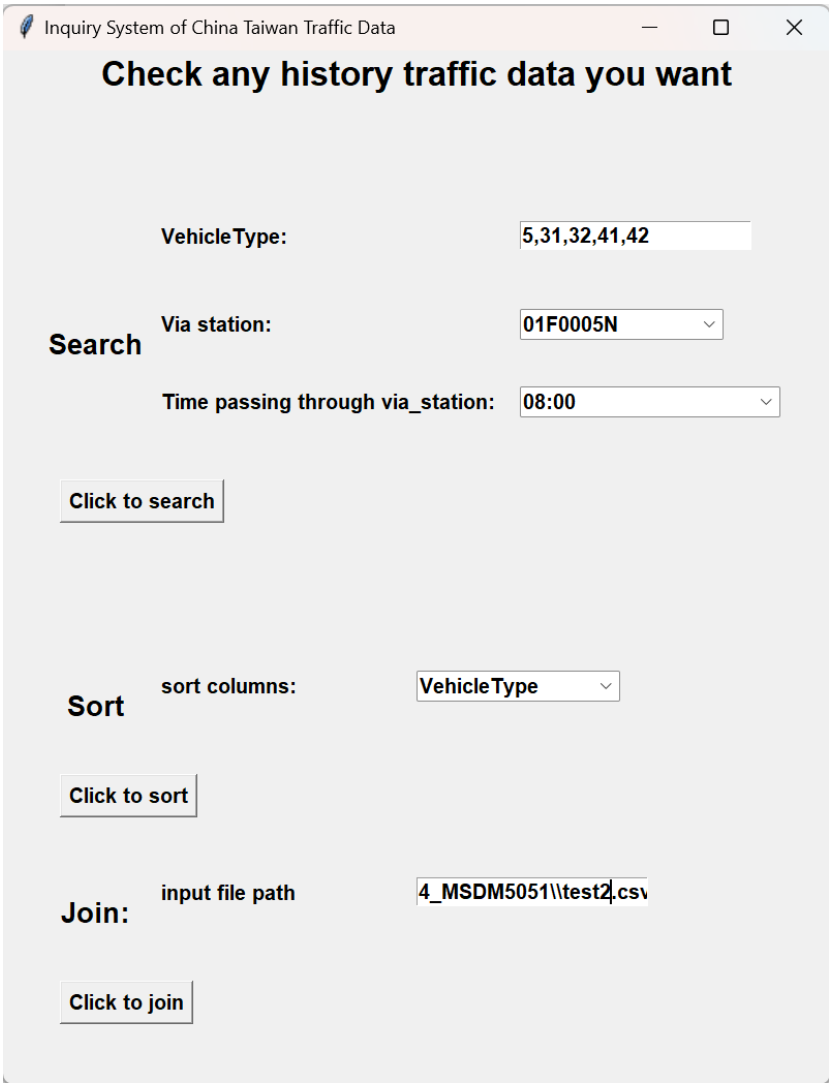
Here's an example using the default inputs for sorting:



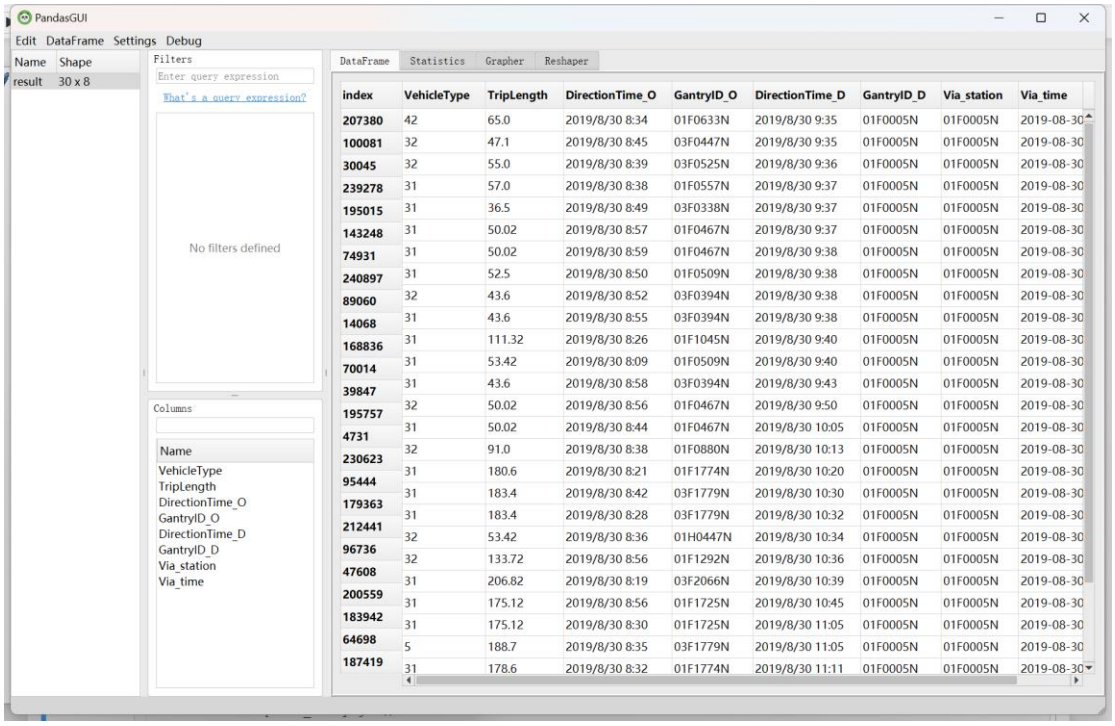
index	VehicleType	TripLength	DirectionTime_O	GantryID_O	DirectionTime_D	GantryID_D	Via_station	Via_time
187419	5	188.7	2019/8/30 8:35	03F1779N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30 11:05:58
4731	31	50.02	2019/8/30 8:44	01F0467N	2019/8/30 10:05	01F0005N	01F0005N	2019-08-30 10:05:12
82504	31	70.2	2019/8/30 8:15	01F0509S	2019/8/30 11:33	01F0005N	01F0005N	2019-08-30 11:33:08
206037	31	178.6	2019/8/30 8:32	01F1774N	2019/8/30 11:11	01F0005N	01F0005N	2019-08-30 11:11:51
64698	31	175.12	2019/8/30 8:30	01F1725N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30 11:05:39
183942	31	175.12	2019/8/30 8:56	01F1725N	2019/8/30 10:45	01F0005N	01F0005N	2019-08-30 10:45:48
200559	31	206.82	2019/8/30 8:19	03F2066N	2019/8/30 10:39	01F0005N	01F0005N	2019-08-30 10:39:03
212441	31	183.4	2019/8/30 8:28	03F1779N	2019/8/30 10:32	01F0005N	01F0005N	2019-08-30 10:32:10
179363	31	183.4	2019/8/30 8:42	03F1779N	2019/8/30 10:30	01F0005N	01F0005N	2019-08-30 10:30:20
95444	31	180.6	2019/8/30 8:21	01F1774N	2019/8/30 10:20	01F0005N	01F0005N	2019-08-30 10:20:49
170450	31	376.4	2019/8/30 8:38	01F3736N	2019/8/30 12:25	01F0005N	01F0005N	2019-08-30 12:25:01
39847	31	43.6	2019/8/30 8:58	03F0394N	2019/8/30 9:43	01F0005N	01F0005N	2019-08-30 09:43:02
112274	31	332.7	2019/8/30 8:34	01F3286N	2019/8/30 13:52	01F0005N	01F0005N	2019-08-30 13:52:12
74931	31	50.02	2019/8/30 8:59	01F0467N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:01
168836	31	111.32	2019/8/30 8:26	01F1045N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30 09:40:09
14068	31	43.6	2019/8/30 8:55	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:56
240897	31	52.5	2019/8/30 8:50	01F0509N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:03
239278	31	57.0	2019/8/30 8:38	01F0557N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:26
70014	31	53.42	2019/8/30 8:09	01F0509N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30 09:40:25
143248	31	50.02	2019/8/30 8:57	01F0467N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:54
195015	31	36.5	2019/8/30 8:49	03F0338N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30 09:37:32
146701	32	301.1	2019/8/30 8:43	03F2985N	2019/8/30 11:37	01F0005N	01F0005N	2019-08-30 11:37:42
100081	32	47.1	2019/8/30 8:45	03F0447N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30 09:35:49
30045	32	55.0	2019/8/30 8:39	03F0525N	2019/8/30 9:36	01F0005N	01F0005N	2019-08-30 09:36:01
195757	32	50.02	2019/8/30 8:56	01F0467N	2019/8/30 9:50	01F0005N	01F0005N	2019-08-30 09:50:05
96736	32	53.42	2019/8/30 8:36	01H0447N	2019/8/30 10:34	01F0005N	01F0005N	2019-08-30 10:34:18
89060	32	43.6	2019/8/30 8:52	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30 09:38:15
230623	32	91.0	2019/8/30 8:38	01F0880N	2019/8/30 10:13	01F0005N	01F0005N	2019-08-30 10:13:16
47608	32	133.72	2019/8/30 8:56	01F1292N	2019/8/30 10:36	01F0005N	01F0005N	2019-08-30 10:36:08
	42	65.0	2019/8/30 8:34	01F0633N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30 09:35:08

3.3. 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:



index	VehicleType	TripLength	DirectionTime_O	GantryID_O	DirectionTime_D	GantryID_D	Via_station	Via_time
207380	42	65.0	2019/8/30 8:34	01F0633N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30
100081	32	47.1	2019/8/30 8:45	03F0447N	2019/8/30 9:35	01F0005N	01F0005N	2019-08-30
30045	32	55.0	2019/8/30 8:39	03F0525N	2019/8/30 9:36	01F0005N	01F0005N	2019-08-30
239278	31	57.0	2019/8/30 8:38	01F0557N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30
195015	31	36.5	2019/8/30 8:49	03F0338N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30
143248	31	50.02	2019/8/30 8:57	01F0467N	2019/8/30 9:37	01F0005N	01F0005N	2019-08-30
74931	31	50.02	2019/8/30 8:59	01F0467N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30
240897	31	52.5	2019/8/30 8:50	01F0509N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30
89060	32	43.6	2019/8/30 8:52	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30
14068	31	43.6	2019/8/30 8:55	03F0394N	2019/8/30 9:38	01F0005N	01F0005N	2019-08-30
168836	31	111.32	2019/8/30 8:26	01F1045N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30
70014	31	53.42	2019/8/30 8:09	01F0509N	2019/8/30 9:40	01F0005N	01F0005N	2019-08-30
39847	31	43.6	2019/8/30 8:58	03F0394N	2019/8/30 9:43	01F0005N	01F0005N	2019-08-30
195757	32	50.02	2019/8/30 8:56	01F0467N	2019/8/30 9:50	01F0005N	01F0005N	2019-08-30
4731	31	50.02	2019/8/30 8:44	01F0467N	2019/8/30 10:05	01F0005N	01F0005N	2019-08-30
230623	32	91.0	2019/8/30 8:38	01F0880N	2019/8/30 10:13	01F0005N	01F0005N	2019-08-30
95444	31	180.6	2019/8/30 8:21	01F1774N	2019/8/30 10:20	01F0005N	01F0005N	2019-08-30
179363	31	183.4	2019/8/30 8:42	03F1779N	2019/8/30 10:30	01F0005N	01F0005N	2019-08-30
212441	31	183.4	2019/8/30 8:28	03F1779N	2019/8/30 10:32	01F0005N	01F0005N	2019-08-30
96736	32	53.42	2019/8/30 8:36	01H0447N	2019/8/30 10:34	01F0005N	01F0005N	2019-08-30
47608	32	133.72	2019/8/30 8:56	01F1292N	2019/8/30 10:36	01F0005N	01F0005N	2019-08-30
200559	31	206.82	2019/8/30 8:19	03F2066N	2019/8/30 10:39	01F0005N	01F0005N	2019-08-30
183942	31	175.12	2019/8/30 8:56	01F1725N	2019/8/30 10:45	01F0005N	01F0005N	2019-08-30
64698	31	175.12	2019/8/30 8:30	01F1725N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30
187419	5	188.7	2019/8/30 8:35	03F1779N	2019/8/30 11:05	01F0005N	01F0005N	2019-08-30
	31	178.6	2019/8/30 8:32	01F1774N	2019/8/30 11:11	01F0005N	01F0005N	2019-08-30

4. 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','Tri

pLength','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','Tri

pLength','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','Tri
pLength','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()
```